Sparse inference using approximate message passing

Michael Riis Andersen

Kongens Lyngby 2014

DTU Compute, Technical University of Denmark
Matematiktorvet, building 303B,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3351
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

This thesis deals with probabilistic methods for finding sparse solutions to ill-posed linear inverse problems using both the single measurement vector (SMV) formulation and multiple measurement vector (MMV) formulation. Specifically, this thesis investigates the novel algorithm called *approximate message passing* (AMP) and its Bayesian generalization called *generalized approximate message passing* (GAMP). The AMP algorithm was initially proposed by Donoho et al. (2009) to solve the linear inverse problem in the context of compressed sensing and later generalized by Rangan (2010). Both algorithms are based on approximations of sum-product algorithm formulated for factors graphs. Through numerical simulations, it is verified that the AMP algorithms are able to achieve superior performance in terms of the sparsity under-sampling trade-off for the basis pursuit problem, while maintaining a low computational complexity. Moreover, the GAMP framework is combined with the sparsity promoting spike and slap prior to derive an inference algorithm for the inverse problem in the SMV formulation. This algorithm is extended to the MMV formulation, which allows the use of multiple measurement vectors by imposing a common sparsity pattern on the measurement vectors. This method, which is coined AMP-MMV, provides soft estimates of both the solution and the underlying support. A thorough numerical study verifies the benefits of the MMV formulation and compares it to state-of-the-art methods. This comparison shows that the AMP-MMV is able to match the recovery capabilities of these methods, while maintaining a computational complexity that is linear in all problem dimensions. Yet another model is considered, the AMP-DCS model, which relaxes the assumption of the common sparsity pattern by assuming the sparsity pattern is slowly changing over time according to a binary Markov chain. By means of numerical experiments, it is demonstrated that this approach is preferable to the SMV approach. For automatic tuning of the hyperparameters, Expectation-Maximization (EM) algorithms are derived for both the SMV and MMV formulation.

Keywords: *Inverse problems, Bayesian inference, AMP, GAMP, MMV, sparsity, message passing, sum-product algorithm, factor graphs*

Summary (Danish)

I denne afhandling undersøges statistiske metoder til løsning af underbestemte lineære ligningssystemer ved brug af både single measurement vector (SMV) modellen og multiple measurement vector (MMV) modellen. I afhandlingen undersøges en ny algoritme kaldet *approximate message passing* (AMP), og dens Bayesianske videreudvikling kaldet *generalized approximate message passing* (GAMP). AMP algoritmen er oprindeligt udviklet af Donoho et al. (2009) i forbindelse med compressed sensing problemet og er senere generaliseret af Rangan (2010). Begge algoritmer er baseret på approksimationer til sum-produkt algoritmen formuleret for faktor grafer. Ved hjælp af numeriske simuleringer verificeres det, at AMP algoritmen opnår overlegne resultater ift. undersampling og sparsity niveau, men stadig bibeholder en lav beregningsmæssig kompleksitet. Ved at kombinere GAMP algoritmen med en Bernouilli-Gaussisk a priori fordeling udledes en algoritme til løsning af de før omtalte ligningssystemer. Denne algoritme udvides også til MMV modellen, hvilket gør det muligt at bruge flere observationsvektorer under antagelsen om konstant support. Den udvidede metode, som kaldes AMP-MMV, kan både estimere selve løsningen, men også den bagvedliggende support. Fordelene ved MMV modellen ift. SMV modellen eftervises via numeriske experimenter. Disse numeriske eksperimenter sammenligner også AMP-MMV algoritmen med Bayesianske state-of-the-art metoder på en systematisk måde. Denne sammenligning viser, at AMP-MMV algoritmen kan opnå sammenlignelige resultater med state-of-the-art metoderne, men med en signifikant lavere beregningsmæssig kompleksitet. Ydermere introduceres modellen AMP-DCS, som antager at den bagvedliggende support for løsningerne ændrer sig langsomt som funktion af tid. I AMP-DCS algoritmen modelleres denne antagelse med en binær Markov kæde. Modellen testes ved hjælp af omfattende numeriske simuleringer og det eftervises at overlegne resultater kan opnås med AMP-DCS modellen fremfor SMV tilgangen. Slutteligt beskrives Expectation-Maximization (EM) algoritmer til at estimere hyperparametrene for de omtalte modeller.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Mathematical Modelling and Computation.

This thesis deals with probabilistic methods for solving the linear inverse problem. In particular, the *approximate message passing* framework and its generalization are derived in detail. These derivations involve a quite high number of equations and a lot of details and therefore some parts of this thesis might be a bit "heavy" reading. But it is the hope, that all these details might make these algorithms more transparent and accessible. Besides the derivations, this thesis also provides a thorough numerical study of the properties of these algorithms.

The project was carried out from September 2013 to February 2014.

Lyngby, 03-February-2014

Michael Riis

Michael Riis Andersen
s112386

Acknowledgements

I would like to thank my supervisors Professor Lars Kai Hansen and Associate Professor Ole Winther at Department of Applied Mathematics and Computer Science for giving me competent and invaluable guidance. I especially would like to thank Lars Kai for giving me opportunities and for introducing me to the fascinating field of inverse problems.

I would also like to thank Rikke Bech Espersen for her patience and for proof reading this thesis.

Abbreviations

AMP	Approximate message passing
ARD	Automatic relevance determination
BG	Bernoulli Gaussian
BP	Basis Pursuit problem
BPDN	Basis Pursuit De-noising problem
DCS	Dynamic compressed sensing
EEG	Electroencephalography
EM	Expectation-Maximization
FOCUSS	Focal underdetermined system solver
GAMP	Generalized approximate message passing
I.I.D.	Independent and identically distributed
LASSO	Least absolute shrinkage and selection operator
MAP	Maximum a posterior
MMSE	Minimum mean squared error
MMV	Multiple measurement vector
SBL	Sparse bayesian learning
SMV	Single measurement vector
SNR	Signal-to-noise ratio
VG	Variational garrote

Nomenclature

$\boldsymbol{A}^T$	Transpose of $\boldsymbol{A}$
$\boldsymbol{x}^k$	Value of $\boldsymbol{x}$ in the k'th iteration
m	Number of equations/rows
n	Number of unknowns/columns
k	Number of non-zero elements in vector
δ	Measure of under-determinacy of a linear system, defined as $\delta = m/n$.
ρ	Measure of sparsity, defined as $\rho = k/m$
$[m]$	Set of integers from 1 to m, i.e. $[m] = \{i \,\vert\, i \in \mathbb{N}, 1 \leq i \leq m\}$.
$\langle \boldsymbol{x} \rangle$	Average of vector $\boldsymbol{x}$
$\boldsymbol{I}_m$	$m \times m$ identity matrix
$\mathcal{N}\left(x \vert \mu, \sigma^2\right)$	Probability density function of Gaussian distribution with mean μ and variance σ^2 evaluated at x.
$\mathcal{N}\left(\mu, \sigma^2\right)$	Gaussian distributed random variable with mean μ and variance σ^2
$\mathbb{E}\left[X\right]$	Expected value of random variable X
$\mathbb{V}\left[X\right]$	Variance of random variable X
$\equiv$	Denotes an identity or definition
$\mathbb{I}\left[a\right]$	Indicator function for proposition a

Contents

Chapter 1

Introduction

In the last two decades, sparsity and sparse models have been experiencing a great increase in interest from the machine learning and signal processing communities [Ela12] and the list of successful applications is numerous. One of the successful applications is the problem of computing sparse solutions to linear inverse problems, which is the topic of this thesis. Problems of this type arise in many different applications in both science and engineering. A non-exhaustive list of examples includes imaging problems [RS08], band-limited extrapolation [CP91], stock market analysis [RZ96] and deconvolution or deblurring of images [OBDF09].

In general, inverse problems refer to the task of the inferring the state of a system that is not directly observable, but only available through a set of measurements. These types of problems are often ill-posed, i.e. not well-posed, and according to the french mathematician Jacques Hadamard, a well-posed problem is defined as a problem for which:

1. a solution exists (existence)
2. the solution is unique (uniqueness)
3. the solution changes smoothly w.r.t. the initial conditions (stability)

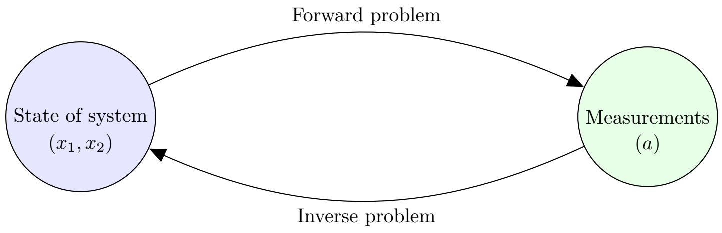


Figure 1.1: Illustration of the relation between the forward and inverse problem.

If a problem does not fulfil these three properties, it is ill-posed by definition.

These concepts are illustrated using the very simple task of computing the arithmetic average of the two numbers x_1 and x_2. This problem is clearly well-posed according to the Hadamard-definition, since a unique solution exists: $a = \frac{1}{2}(x_1 + x_2)$ and the solution is continuous w.r.t. both x_1 and x_2. If we denote the problem of computing the average of the two numbers as the *forward problem*, we can now consider the *inverse problem*. That is, given the arithmetic average a, compute the two numbers x_1 and x_2, where (x_1, x_2) is now considered the state of the system and a is the measurement, see figure 1.1. This inverse problem has a solution, but it is not unique. In fact, every tuple of real numbers (x_1, x_2), which satisfies the relation $x_1 = 2a - x_2$, is a solution. Therefore, the particular problem of inferring (x_1, x_2) from a is a very simple example of an ill-posed inverse problem. In general, given the state of the system it is relative easy to compute the measurements, whereas solving the inverse problem is usually much more difficult.

1.1 Sparse Solutions to Linear Inverse Problems

The aim of this thesis is to investigate the framework of *approximate message passing* [DMM09] for finding sparse solutions to linear inverse problems. This work is mainly motivated by problem of *source localization* for electroencephalography (EEG) [BML01, NS06], which aims to localize the sources of neural electrical activity in the human brain using non-invasive measurements of electromagnetic signals. Thus, the distribution of the neural electrical brain activity is the desired state and the electromagnetic signals are the measurements (see appendix D for more details). However, the methods and algorithms discussed in this thesis apply to ill-posed linear inverse problems in general.

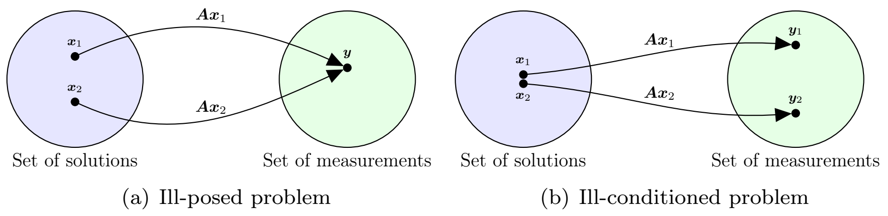


Figure 1.2: (a) Illustration of an ill-posed problem, where more than one solution map to the same set of measurements. (b) Illustration of the concept of an ill-conditioned problem, where two similar solutions map to very different measurements.

Formally, a linear inverse problem has the following form:

$$\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{e}, \tag{1.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ is a vector of m measurements, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is the so-called *forward matrix*, $\boldsymbol{e} \in \mathbb{R}^m$ is a corruptive noise vector and $\boldsymbol{x} \in \mathbb{R}^n$ is the desired state of the system. The objective is then to reconstruct $\boldsymbol{x}$ from the pair $(\boldsymbol{A}, \boldsymbol{y})$. From this formulation, it is immediately seen that the (noiseless) measurements are easily computed based on knowledge of the forward matrix $\boldsymbol{A}$ and the true solution $\boldsymbol{x}$.

For many problems of practical interest (including EEG source localization), the number of measurements are much smaller than the dimension of the desired state vector $\boldsymbol{x}$, i.e. $m << n$, and this effectively makes the linear system of equations in eq. (1.1) under-determined. This implies that, if a solution exists, then it is not unique and therefore, the problem is indeed ill-posed according to the Hadamard definition.

Besides being ill-posed, many linear inverse problems also provide another difficult challenge: they are *ill-conditioned*, meaning that the solution is highly sensitive to small perturbations in the measurements. This is of course an undesirable characteristic for a noisy system. The ”degree” of illconditioned-ness can be measured by the *condition number*[1] of the forward matrix $\boldsymbol{A}$, where a high condition number implies that the problem is ill-conditioned. Figure 1.2 illustrates the differences between a problem being ill-posed and being ill-conditioned.

Because of these issues, additional assumptions or constraints have to be imposed on the system in eq. 1.1 in order for it to be solved. Put another way, the

[1]The condition number of a matrix $\boldsymbol{A}$ is the ratio of the largest and smallest singular value of $\boldsymbol{A}$.

systems needs to be regularized, where the classical approach is Tihkonov regularization, in which solutions with smaller norms are preferred [TA77, McD09]. Another approach is to assume that the desired solution $\boldsymbol{x}$ is *sparse* [Ela12], which is the approach taken in this thesis.

So what does it mean for $\boldsymbol{x}$ to be sparse and how does it help? It means that most of the entries in $\boldsymbol{x} \in \mathbb{R}^n$ is zero, or put another way: the energy of $\boldsymbol{x}$ is concentrated in a small number of positions. The vector $\boldsymbol{x}$ is said to be k-sparse, if it has at most $k \leq n$ non-zero entries. In the noiseless case, i.e. $\boldsymbol{e} = 0$, and under certain assumptions on the forward matrix $\boldsymbol{A}$, it has been shown that if $\boldsymbol{x}$ is *sufficiently* sparse compared to the undersamplingsratio, i.e. the ratio $\frac{m}{n}$, then exact recovery of $\boldsymbol{x}$ is possible [CRT06, DT10]. Informally, the higher degree of sparsity, i.e. the smaller k, the fewer samples are required to achieve perfect reconstruction. This is referred to as the sparsity-undersampling tradeoff. Besides mathematical convenience, sparsity also has the advantage that sparse models are usually easier to interpret than fully ”dense” models. But it is important to stress that the location of the non-zero elements in $\boldsymbol{x}$, also known as the support of $\boldsymbol{x}$, are usually not known in advance.

We now return to the simple problem of recovering (x_1, x_2) from the arithmetic average a, for which we concluded an infinite number of solutions existed. Suppose now that the desired solution (x_1, x_2) is 1-sparse, then it is easily seen that $(2a, 0)$ and $(0, 2a)$ are the only solutions to the system. Thus, the solution is still not unique, but the infinite set of solutions is effectively reduced to a set of only 2 solutions.

However, many natural signals of interest do not have an exact sparse representation, especially not under contamination of noise. But it turns out that many signals can be well approximated by a sparse signal when represented in a suitable basis. For instance, a signal dominated by a single sine-wave is well approximated using a 1-sparse signal in suitable Fourier basis, many natural images can be well approximated by a sparse signal using a Wavelet or Curvelet representation [SFM10] and so on. Signals, which are well approximated by a sparse signal in a known basis, are referred to as *compressible* signals.

A natural generalization of the linear inverse problem in eq. (1.1) is the so-called *multiple measurement vector problem (MMV)* [CREKD05], where, as the name suggests, multiple measurement vectors $\{\boldsymbol{y}^t\}$ are available at consecutive time steps $t = 1, .., T$:

$$\boldsymbol{y}^t = \boldsymbol{A}^t \boldsymbol{x}^t + \boldsymbol{e}^t, \tag{1.2}$$

In the context of the MMV problem, the formulation of the linear inverse problem in eq. (1.1) is referred to as a *single measurement vector problem (SMV).*

Clearly, the MMV problem can be approached by solving T individual (SMV) problems and nothing is gained. On the other hand, if the measurement vectors $\{\boldsymbol{y}^t\}_{t=1}^T$ are obtained within a small period of time, it is often reasonable to assume that support of the solutions $\boldsymbol{x}^t$ is constant w.r.t. time, i.e., the locations of the non-zero elements in $\boldsymbol{x}^t$ are the same for all $t = 1, .., T$. This is often referred to as the *common sparsity assumption* [CREKD05] or *joint sparsity assumption* [vdBF10]. In this case, the estimates of the solution vectors $\{\boldsymbol{x}^t\}_{t=1}^T$ can be greatly improved by using multiple measurement vectors [ZR13, WR07]. However, note that index t does not necessarily have to be a time index. That is, the evolution of the measurements $\{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T\}$ does not necessarily have to be temporal, it can also be spatial etc. as long as it fits into the formulation in eq. (1.2) and satisfies the common sparsity assumption. Here a temporal evolution is assumed without loss of generality. A few examples of applications of the MMV formulation are face recognition from video [MW12], through-the-wall radar imagery [BTA11] and EEG-based source localization [ZR13].

By assuming the forward matrix $\boldsymbol{A}$ does not change with time index t, and by concatenating the measurement vectors into a matrix, i.e. $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & .. & \boldsymbol{y}_T \end{bmatrix} \in \mathbb{R}^{m \times T}$, and similar for the solution vectors $\{\boldsymbol{x}^t\}_{t=1}^T$ and error vectors $\{\boldsymbol{e}^t\}_{t=1}^T$ the MMV problem can be reformulated using matrices:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{E}, \tag{1.3}$$

where $\boldsymbol{Y} \in \mathbb{R}^{m \times T}$ is the measurement matrix, $\boldsymbol{E} \in \mathbb{R}^{m \times T}$ is the error matrix and $\boldsymbol{X} \in \mathbb{R}^{n \times T}$ is the desired solution matrix. Using this formulation, the common sparsity assumption is then manifested as *row sparsity* of $\boldsymbol{X}$. The non-zero rows of $\boldsymbol{X}$ are referred to as the true signals or sources .

Consider the case, where multiple measurement vectors are available, but the underlying sparsity pattern is changing with time, i.e. the common sparsity assumption is violated. If the sparsity patterns at two subsequent time steps are independent, then the problem should be approached as independent SMV problems. On the other hand, if the sparsity pattern is changing slowly with time, it can be incorporated into the model. Such models are referred to as *MMV with dynamic support* as opposed to *MMV with static support*. The use of such models can be justified for many applications. For instance, this type of model is appropriate for EEG source localization due to the dynamic nature of the human brain [NS06, AM12].

1.2 Literature Review

The purpose of this section is to give an overview of the literature regarding the SMV and MMV problems. This section also serves to introduce some of the terminology used in this thesis. First, sparse methods for solving the linear inverse problem are discussed followed by a short review of methods for the MMV extension.

In the noiseless case, the linear inverse problem and its variants are often referred to as a *basis pursuit problem*, since the formulation is equivalent to finding the best representation $\boldsymbol{x}$ of a signal $\boldsymbol{y}$ in an over-complete dictionary $\boldsymbol{A}$ [CDA08]. Similarly, the noisy version of the linear inverse problem is often referred to as the *basis pursuit de-noising problem.* Also note, that a linear inverse problem of the form in eq. 1.1 can equivalently be cast as a linear regression problem, where $\boldsymbol{A}$ is the design matrix and $\boldsymbol{x}$ are the parameters or weights to be estimated. Thus, the entire machinery of statistics and machine learning utilized to solve problems of this form.

There exist a number of different approaches to the BP and BPDN problem, where some of the major classes of methods are:

1. Pursuits methods / greedy methods
2. Optimization-based methods
3. Probabilistic methods
4. Brute force/combinatorial serach

Pursuit methods are greedy methods in the sense that they choose the most beneficial step in each iteration. That is, the estimated solution $\hat{x}$ is iteratively improved by adapting the change in one or more components of solution that yields the greatest improvement of the estimate according to some measure. One of the simplest pursuit algorithms is the *orthogonal matching pursuit*-method (OMP) [DMA97].

Starting from an empty solution, i.e. $\hat{x} = 0$, OMP iteratively identifies the feature, which shows strongest correlation with the residuals and then adds this feature to the solution. The estimate $\hat{x}$ is then updated in a least squares fashion using the included features, after which the residuals are updated using the new estimate. These steps are repeated until a stopping criteria is met. Several extensions have been suggested for this method, i.e. the *compressive sampling* *matching pursuit* (CoSaMP) method [NT10], which allows the inclusion of multiple features as well as pruning of features in each iteration.

The pursuit methods are closely related to the *iterative thresholding methods* [MD10], which can be subdivided into *soft* or *hard* thresholding methods. Both types of methods alternates between steps of the form:

- Compute residuals: $\boldsymbol{r}_k = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_k$
- Update estimate: $\boldsymbol{x}_{k+1} = h(\boldsymbol{x}_k + \kappa \cdot \boldsymbol{r}_k)$,

where $\boldsymbol{r}_k$ is an estimate of the current residual, the function h determines the thresholding policy and κ controls the step-size. These types of methods are in general very simple and therefore also fast, but at the cost of suboptimal performance in terms of the sparsity-undersampling trade-off [MD10].

We now turn the attention towards the optimization-based methods. Most of the optimization-based methods can be divided into two stages. In the first stage an optimization objective or cost function is designed to encourage the desired behaviour of the solution, i.e. sparsity, temporal smoothness, spatial smoothness and so on. In the second stage, either an existing optimization scheme (line search, trust region etc.) is applied to the cost function or a new optimization scheme is designed specifically to the given cost function.

For a vector $\boldsymbol{x} \in \mathbb{R}^n$, the pseudonorm[2] or counting function $||\cdot||_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ returns the number of non-zero elements and thus, is useful for describing the degree of sparsity for a vector, i.e. of a vector $\boldsymbol{x}$ is k-sparse if and only if $||\boldsymbol{x}||_0 \leq k$.

In the search for sparse solutions, it is indeed tempting to try to minimize $||\boldsymbol{x}||_0$ subject to a data fitting constraint, i.e. a least squares criterion:

$$\min_{\boldsymbol{x}} ||\boldsymbol{x}||_0 \qquad \text{s.t.} \qquad ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 \leq \epsilon, \quad 0 < \epsilon, \tag{1.4}$$

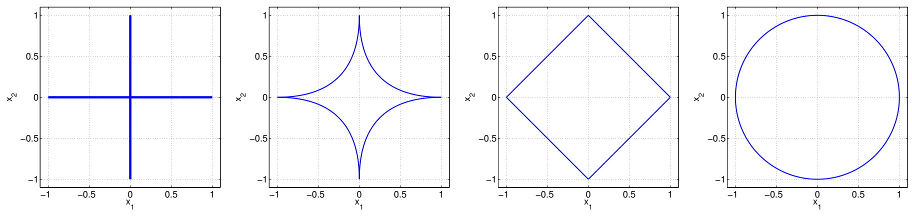
or one of the related problems

$$\min_{\boldsymbol{x}} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 \qquad \text{s.t.} \qquad ||\boldsymbol{x}||_0 \leq s, \tag{1.5}$$

$$\min_{\boldsymbol{x}} \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 \qquad \text{s.t.} \qquad \lambda ||\boldsymbol{x}||_0 , \tag{1.6}$$

However, these objective functions are both highly non-convex and non-smooth and are therefore hard to optimize. One approach to circumvent this, is the

[2]It is not an actual norm, since $||\alpha\boldsymbol{x}||_0 \neq |\alpha| \cdot ||\boldsymbol{x}||_0$ for $\alpha \in \mathbb{R}$



(a) $p = 0$ (b) $p = 0.5$ (c) $p = 1$ (d) $p = 2$

Figure 1.3: Boundary of the unit balls for $||\cdot||_p$ for $p = \{0, 0.5, 1, 2\}$. Note, how norm for $p \geq 1$ can be considered as convex approximations to the ℓ_0-pseudonorm.

convex relaxation approach, where the non-convex optimization objective is approximated by a convex function (see figure 1.3). The LASSO [Tib96] problem is an example of such an approach and is given by:

$$\min_{\boldsymbol{x}} \frac{1}{2} ||\boldsymbol{Ax} - \boldsymbol{y}||_2^2 + \lambda ||\boldsymbol{x}||_1 , \tag{1.7}$$

where λ is a hyperparameter controlling the trade-off between sparsity and fidelity of the fit. This is in general a robust approach, but it requires tuning the regularization parameter λ. However, the closely related Least-angle Regression-algorithms provides a fast way of computing the entire regularization path [EHJT].

The *minimum norm*-methods are a subclass of the optimization-based methods. In the noiseless case, the minimum 2-norm solution is simply the solution, which minimizes to ℓ_2-norm subject to the constraint $\boldsymbol{y} = \boldsymbol{Ax}$. Several improvements to this approach have been proposed. One is the re-weighted minimum norm algorithm, which is also known as the FOCUSS[3] algorithm [GR97]. Using an initial estimate, the FOCUSS algorithm iteratively minimizes the re-weighted norm $\sum_{i=1,w_i\neq 0}^{n} \left(\frac{x_i}{w_i}\right)^2$ subject to the constraint $\boldsymbol{y} = \boldsymbol{Ax}$, where w_i is the value of x_i in the previous iteration. This approach produces sparse solutions, because if the i'th component of $\boldsymbol{x}$ is small in the k'th iteration, it is even smaller in iteration $k + 1$ and so on. These methods can also be extended to the noisy formulation, by introducing a regularization parameter and changing the hard data constraint with a soft constraint [REC+10].

We now turn the attention to the class of probabilistic models. Most methods in this class can also be divided into two stages. In the first stage a probabilistic

[3] FOcal Underdertermined System Solver

model is set up by introducing to the necessary random variables and by specifying the relationship between them and in the second stage, an inference method is applied to the model. It is worth noting that many algorithms can be derived from more than one perspective, i.e. the LASSO [Tib96] can both be derived from the optimization perspective as well as from the probabilistic perspective. Sometimes it is even beneficial to view a method from another perspective.

For the probabilistic models, we will mainly focus on Bayesian models, since they allow the sparsity assumption to be incorporated using suitable prior distributions. In particular, a number of methods are based on the Bayesian framework coined *Automatic Relevance Determination* (ARD) [Nea95, WN09] or on the related *Sparse Bayesian Learning* (SBL) framework [Tip01]. In the ARD framework, a Gaussian noise distribution is combined with a zero-mean Gaussian prior on $\boldsymbol{x}$, which is imposed to regularize the estimated solution. This prior assigns individual variance hyperparameters toto each element in $\boldsymbol{x}$, i.e. the prior distribution has n individual variance parameters. By marginalizing out the solution $\boldsymbol{x}$, and applying the so-called *evidence approximation* [Bis06], the marginal likelihood function of the measurements $\boldsymbol{y}$ conditioned on the hyperparameters is obtained and optimized with respect to these hyperparameters. During this optimization, the individual variance hyperparameters of irrelevant features will shrink towards 0 and thus effectively prune the corresponding of the parameter of the model.

Another popular Bayesian approach is the use of a Bernoulli-Gaussian prior on each x_i [IR05]. That is, the prior distribution on $\boldsymbol{x}$ is assumed to be a mixture of a Dirac delta function at zero and a Gaussian distribution. Under this prior, x_i has point mass at $x_i = 0$ and therefore this is a sparsity promoting prior. Because of the form of the density, this type of prior is also known as a ”spike and slap”-prior.

The Variational Garrote (VG) [KG12] uses a prior similar to the ”spike and slap”-prior, but instead the support variables are marginalized out and the posterior distribution is obtained using a mean-field variational approximation [Bis06]. In [KG12], a single hyperparameter is controlling the sparsity of the solution. This hyperparameter is tuned using a cross validation procedure. In [AHH13], the VG model is extended to estimate the hyperparameter from the data using an emperical Bayes approach. For more details about the VG approach, see Appendix E.

Many models and algorithms for the SMV formulation have been extended to the MMV formulation. In [CREKD05], Cotter et al. describe natural extensions of the OMP and FOCUSS methods to the MMV formulation. Similarly, many of the probabilistic methods have been extended as well. The SBL method is also extended to the MMV formulaton resulting in the M-SBL [WR07] and

TM-SBL methods [ZR13]. The model M-SBL proposed by Wipf et al. is a straightforward extension to the MMV problem using SBL framework, where each row of the source matrix $\boldsymbol{X}$ shares a common hyperparameter. In fact, Wipf et al. showed that the M-SBL method implicitly minimizes the ℓ_2-norm of each row in $\boldsymbol{X}$. This way the pruning of the features become row-based. In [ZR13], Zhang et al. introduced the TM-SBL model, which is a further extension of the M-SBL model that takes temporal correlation of the non-zero sources into account. The TM-SBL model assumes that each source, i.e. each row of $\boldsymbol{X}$, has the same correlation structure and Zhang et al. argues that TM-SBL only differs from M-SBL by implicitly minimizing the Mahalanobis distance of the rows of $\boldsymbol{X}$ instead of the ℓ_2-norm.

Many researchers have been working on deriving theoretical bounds and guarantees for when the inverse problem is solvable. We will not go into details here, but the interested reader is referred to [EK12] for an overview. However, we will review the concept of *phase space* and *phase transitions* introduced by Donoho et al. [DT10]. Consider a noiseless linear inverse problem $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, then define the *undersamplingsratio* $\delta \in [0, 1]$ as the ratio of measurements and unknowns, i.e. $\delta = m/n$ and the *sparsity* $\rho \in [0, 1]$ as the ratio of non-zero elements and measurements, i.e. $\rho = k/m$. Note also that the reciprocal values of ρ can be interpreted as the number of measurements per parameter. Donoho et al. define the *phase space* as the domain $(\delta, \rho) \in [0, 1]^2$.

Many reconstruction algorithms exhibit a so-called *phase transition* in this space. That is, the recovery performance of a given method partitions the phase space into two regions: a solvable and an unsolvable region. The phase transition curve is then the boundary between these two regions. These curves then provide a neat way of comparing the reconstruction performance for different algorithm.

For a noiseless linear inverse problem with I.I.D. Gaussian forward matrix, the phase transition curve for the LASSO can be computed analytically in large system limit i.e. $n, m \to \infty$ for $m/n \to \delta$ and $k/m \to \rho$ using combinatorial geometry [DT10, DT09, DMM11]. The asymptotic phase transition curve is given by:

$$\rho_{CG}(\delta) = \max_{z \geq 0} \frac{1 - (2/\delta)\left[(1 + z^2)\Phi(-z) - z\phi(z)\right]}{1 + z^2 - 2\left[(1 + z^2)\Phi(-z) - z\phi(z)\right]} \tag{1.8}$$

where $\phi(z)$ and $\Phi(z)$ are the density and cumulative distribution of a standardized Gaussian variable z, respectively. This curve is shown in figure 1.4, where the region below the curve is solvable region. As one moves up and to the left in the phase space, problems become more undersampled and less sparse and therefore more difficult to solve. The curve $\rho_{CG}(\delta)$ thus provides a convenient frame of reference when designing and investigating new methods.

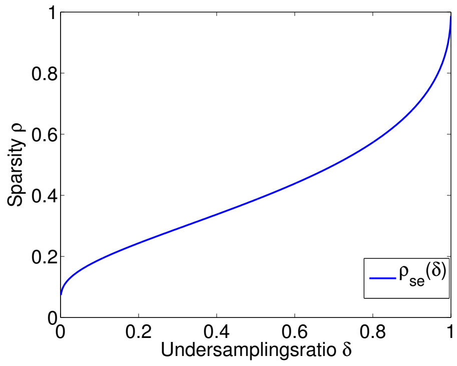


Figure 1.4: Asymptotic phase transition curve for ℓ_1 minimization predicted by combinatorial geometry.

1.3 Problem Definition

As stated earlier, the topic of this thesis is linear inverse problems. Among the many different approaches to these problems, Bayesian methods have been shown to provide state-of-the-art recovery performance compared to other methods. However, a large portion of the Bayesian methods suffer from the fact, that they are inherently slow, which limits their applicability on large-scale problems. On the other end of the spectrum is the class of iterative threshold methods, which are extremely fast but at the cost of suboptimal recovery performance.

But in 2009 David Donoho and his colleagues introduced a framework called *approximate message passing* (AMP) [DMM09], which appear to offer the best from both worlds. That is, AMP provides excellent recovery performance, while maintaining low computational complexity. In 2010, Sundeep Rangan introduced a generalization of this framework, called *Generalized Approximate Message Passing* (GAMP) [Ran10], which allows the use of a broader class of models with the same low computational complexity.

The main goal of this thesis is to analyze and derive the AMP and GAMP frameworks. It is of great interest to investigate how these frameworks can be utilized to construct algorithms, which are capable of doing rapid inference in highly underdetermined noisy systems. The second goal of this thesis is to explore how these algorithms can be extended to the MMV formulation. In particular, it is of interest to investigate the properties of these methods in terms of their sparsity-undersampling trade-off, robustness to noise and computational complexity.

1.4 Thesis Overview

The structure of the thesis is as follows. The thesis consists of 5 chapters (including this introduction) and a number of appendices. Below is a short description of each chapter.

- Chapter 1 is the introduction, including literature review and problem definition.
- Chapter 2 provides a small introduction to message passing algorithm in factor graphs, which also serves to introduce to necessary terminology and notation. The rest of chapter 2 describes the theory for approximate message passing (AMP) and the generalized approximate message passing algorithms (GAMP). Moreover, an inference algorithm (BG-AMP) is derived based on GAMP and the Bernoulli-Gaussian prior.
- Chapter 3 extends the BG-AMP method from chapter 2 into two new algorithms for the multiple measurement vector (MMV) formulation. The first algorithm, AMP-MMV, assumes constant support, whereas the second algorithm, AMP-DCS, assumes slowly changing support.
- Chapter 4 provides an extensive numerical study and discussion of the methods introduced in chapter 2 and 3.
- Chapter 5 summarizes and concludes on the results.

Chapter 2

Theory: The Linear Inverse Problem

The goal of this section is to describe the algorithms *approximate message passing* (AMP) and *generalized approximate message passing* (GAMP). Both of these algorithms are based on message passings techniques in factors graphs. Therefore, the first part of this chapter is devoted to introduce the basic terminology of factor graphs and the associated inference technique called *message passing*. The second part describes the theory behind the AMP algorithm, whereas the third part describes the GAMP algorithm. Finally, the fourth section introduces an algorithm called *BG-AMP*, which is derived using the GAMP framework. Before diving into the theory of these algorithms, we briefly review a few concepts related to basic statistical inference.

The objective is to solve a linear inverse problem of the form:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} \tag{2.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ is a measurement vector, $\boldsymbol{x} \in \mathbb{R}^n$ is the desired solution vector, $\boldsymbol{e} \in \mathbb{R}^m$ is a vector of measurements errors and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is the forward matrix, where it is assumed that $m < n$ such that the system is under-determined.

There are different approaches to statistical inference, where *maximum likelihood estimation (ML)*, *maximum a posteriori estimation estimation (MAP)*, *minimal mean square error estimation* and *Bayesian inference* are commonly used [Bis06]. The *likelihood* of a set of parameters $\boldsymbol{x}$ is the probability of the observed data given the particular set of parameters $\boldsymbol{x}$, or put another way, it is the probability of the data conditioned on the parameters $\boldsymbol{x}$. Now suppose the likelihood is given by $p(\boldsymbol{y}|\boldsymbol{x})$, where $\boldsymbol{y}$ is the observed data and $\boldsymbol{x}$ is the parameters. Then, as the name suggest, the principle of maximum likelihood is to choose the parameters such that the likelihood function is maximized. That is, the ML estimate is given by

$$\hat{\boldsymbol{x}}^{ML} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x}) \tag{2.2}$$

where $\boldsymbol{x}$ is treated as a deterministic variable. Now suppose we treat $\boldsymbol{x}$ as a random variable, then we can assign a *prior distribution* $p(\boldsymbol{x})$ to $\boldsymbol{x}$, which reflects our prior beliefs or assumptions about $\boldsymbol{x}$. For the remainder of this discussion, let $p(\boldsymbol{x}\big|\theta)$ be a parametric distribution, which is parametrized by θ. Since θ is a parameter controlling the prior distribution, it is referred to as a *hyperparameter*

Bayes theorem [Bis06] now allows us to obtain an expression for the posterior distribution of the parameters, i.e. the probability of the parameters $\boldsymbol{x}$ conditioned on the observed values $\boldsymbol{y}$:

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}|\theta)}{p(\boldsymbol{y})}, \tag{2.3}$$

where the term $p(\boldsymbol{y})$ often is called the *evidence* or *marginal likelihood.* Note, that the marginal likelihood is independent of $\boldsymbol{x}$, and therefore

$$p(\boldsymbol{x}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}|\theta) \tag{2.4}$$

Using this result, we can now state the MAP estimate as

$$\hat{\boldsymbol{x}}^{MAP} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}) = \arg\max_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}|\theta) \tag{2.5}$$

where θ is treated a deterministic variable. Thus, the MAP estimator is similar to the ML estimator except for the prior distribution on the parameters. However, the prior distribution can often be interpreted as regularization term and does therefore often have a crucial effect on the final estimate. Next, we consider the concept of *Bayesian inference.*

The basic philosophy of Bayesian inference is that all involved quantities are considered to be random variables. Ideally, we would also assign a prior distribution to the hyperparameter, i.e. $p(\theta)$. In the Bayesian paradigm, we are just as interested in the associated uncertainty as in the estimate itself and therefore we strive to acquire the entire posterior distribution $p(\boldsymbol{x}|\boldsymbol{y})$, and not only the *point estimates* $\hat{x}$ as in ML or MAP estimation. However, for many problems of practical interest, exact inference in the Bayesian paradigm is often infeasible due to analytically intractable integrals and therefore we often have to resort to different approximation schemes.

The last type of inference, we will consider here, is the *minimal mean square error* estimator. That is, the estimator which minimizes the mean error square between the fitted value and the true value. Under mild conditions, the MMSE-estimate is equal to the conditional expectation [Bis06]

$$\hat{x}^{\text{MMSE}}(\boldsymbol{y}) = \mathbb{E}\left[\boldsymbol{x}\middle|\boldsymbol{y}\right] \tag{2.6}$$

We are now ready to dive into the concepts of factor graphs and message passing.

2.1 Inference using Factor Graphs

Most of the methods used in this thesis are based on the so-called message passing techniques [Pea88, Bis06, Bar11] for a graphical representation called a *factor graph* [Loe04, LDH+07, Bis06, Bar11, Mac03] and therefore this section serves to give a brief introduction to the basic methodology and terminology of these concepts.

Informally, a factor graph[1] is a graphical representation of the decomposition of a global function $f(\boldsymbol{x})$ into its local factors $f_a(\boldsymbol{x}_a)$. Consider a function $f(x_1, x_2, x_3)$ and assume it decomposes as follows:

$$f(x_1, x_2, x_3) = f_1(x_1, x_2) \cdot f_2(x_2, x_3) \cdot f_3(x_3) = \prod_a f_a(\boldsymbol{x}_a) \tag{2.7}$$

where $\boldsymbol{x}_a$ is the set of variables associated to the factor $f_a(\cdot)$, i.e. $\boldsymbol{x}_1 = (x_1, x_2)$. Thus, the *global function* $f(x_1, x_2, x_3)$ decomposes into 3 *local functions*, where each local function is a function of a subset of the variables. This thesis deals with probabilistic models, and therefore the global function will be a joint distribution of interest, while the local factors will correspond to marginal and conditional distributions. However, this framework is not restricted to probability distributions.

Formally, a factor graph is a bipartite graph consisting of a set of edges and two disjoint sets of nodes: variables nodes and factor nodes. Each variable node corresponds to a unique variable in the global function and is represented by a circle. Each factor node corresponds to a unique factor function in the

[1] In the literature, there exists different kinds of factor graphs, but here we will adapt the style and notation from [Bis06].

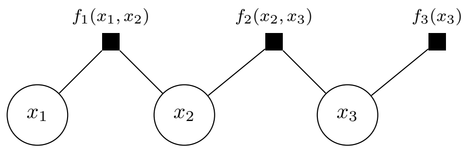


Figure 2.1: Factor graph representation for the decomposition of the global function in eq. (2.7). Each variable is uniquely represented by a variable node (circles) and each factor function is uniquely represented by a factor node (black squares). An edge between a variable node x_j and factor node $f_a(\cdot)$ indicates that the given factor function $f_a(\cdot)$ is a function of x_j.

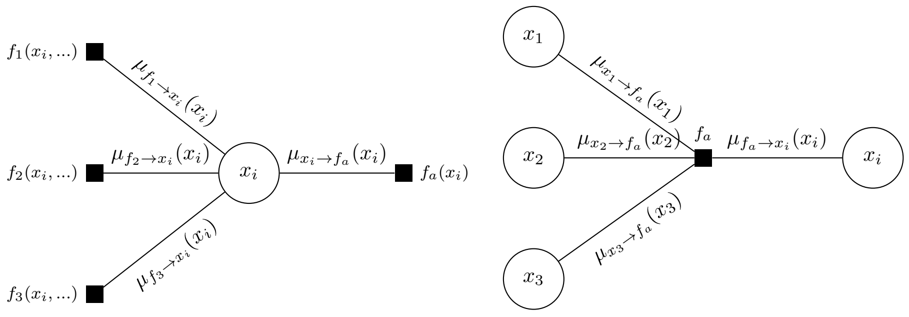
decomposition of the global function and is represented by a filled black square. There is an edge between variable node x_j and a factor node $f_a(\cdot)$ if and only if $f_a(\cdot)$ is a function of x_j. Note, that the edges are undirected. Using these properties it is possible to translate the decomposition of a global function into a factor graph and vice versa.

Figure 2.1 shows the corresponding factor graph representation of the decomposition in eq. (2.7). Since $f_1(x_1, x_2)$ is a function of both x_1 and x_2, there are edges from factor node $f_1(x_1, x_2)$ to variable nodes x_1 and x_2. The factor node $f_3(x_3)$ is only connected to variable node x_3 due the factor function $f_3(x_3)$ only being a function of x_3.

The Sum-Product Algorithm

We will now use the concept of factor graphs to introduce to the so-called *sum-product algorithm* [KFL01, Bis06], which is a computationally efficient algorithm for computing the marginal posterior distribution of any variable in a factor graph. The algorithm was initially introduced in the form of the *Belief propagation* in Bayesian networks [Pea88] by Judea Pearl in 1982, and later generalized to the sum-product algorithm presented here.

The marginal posterior distributions obtained by the sum-product algorithm is only exact when the underlying factor graph is a poly-tree [Bis06]. That is, if the underlying factor graphs contains cycles or loops, the sum-product algorithm is not guaranteed to produce exact inference. Nevertheless, for some cyclic factor graphs the sum-product algorithm can be successfully applied in an iterative manner to obtain approximate posterior distributions [FM98]. We will make extensive use of this fact, when deriving the *approximate message passing* algorithm.



(a) From variable node to factor node (b) From factor node to variable node

Figure 2.2: (a) Illustration of the messages involved in forming the message from variable node x_i to factor node f_a. (b) Illustration of the messages involved in forming the message from factor node f_a to variable node x_i.

The sum-product algorithm for factor graphs essentially boils down to a set of rule for propagating local "messages" between the variable nodes and factor nodes. It is out of the scope of this thesis to derive and prove the sum-product algorithm, instead we simply state the resulting expression, but the derivations can be found [Bis06, ch. 8]. First the necessary notation is introduced. The sum-product algorithms involves two types of messages: messages from variable nodes to factor nodes and messages from factor nodes to variables. Let $\mu_{x_i\to f_a(\boldsymbol{x}_a)}(x_i)$ denote the local message sent from variable node x_i to factor node $f_a(\boldsymbol{x}_a)$ and similarly, let $\mu_{f_a(\boldsymbol{x}_a)\to x_i}(x_i)$ denote the local message sent from factor node $f_a(\boldsymbol{x}_a)$ to variable node x_i. Both types of messages are functions of the involved variable and the involved variable only. Furthermore, let $\text{ne}(x)$ be the set of neighbouring nodes at node x, e.g. $\text{ne}(f_2) = \{x_2, x_3\}$ and $\text{ne}(x_3) = \{f_2, f_3\}$ for the factor graph in figure 2.1.

In the following it is assumed that all variables are discrete, but sum-product algorithm applies equally well to continuous variable by exchanging the sums with suitable integrals. Equipped with this notation we can now define these messages. The message sent from non-leaf variable node x_i to factor node f_a is given by

$$\mu_{x_i\to f_a}(x_i) = \prod_{b\in ne(x_i)\setminus f_a} \mu_{f_b\to x_i}(x), \tag{2.8}$$

That is, the message sent from x_i to f_a is simply the product of incoming messages at node x_i, except the message from the target node f_a. This is illustrated in figure 2.2(a). From this definition, it is seen that if a variable node only has two neighbours it simply "forwards" the incoming messages. If the variable node x_i is a leaf node, the message simply becomes the unit function:

$$\mu_{x_i \to f_a}(x_i) = 1; \tag{2.9}$$

The other type of message is from a non-leaf factor node f_a to a variable node x_i and is given by:

$$\mu_{f_a \to x_i}(x_i) = \sum_{x \forall j \neq i} \left[f_a(x_i, ...) \prod_{j \in ne(x) \setminus x_i} \mu_{x_j \to f_a}(x_j) \right] \tag{2.10}$$

where the sum is over all involved variables except x_i. That is, the message from factor node f_a to variable node x_i is given by the sum over the product of the factor function f_a itself and all the incoming messages, except the messages from the target variable node x_i. If the factor node f_a is a leaf-node, the message simply becomes the factor function itself:

$$\mu_{f_a \to x_i}(x_i) = f_a(x_i) \tag{2.11}$$

Note, that the message associated with an edge from variable x_i to factor f_a (and vice versa) is always a function of x_i and only x_i. Furthermore, due to the recursive nature of the messages, a given node is only "ready" to send its message if it has received all the necessary incoming messages first, which explains why the underlying factor graph must have tree structure. Otherwise, there would be a "circular dependency".

Using the above expression for the messages, the main results can now be stated as in Table 2.1.

Table 2.1: Sum-product rules for obtaining posterior distributions.

1. The (unnormalized) marginal posterior distribution of any variable x_i conditioned on the set of observed variables, is obtained by computing the product of all incoming messages at node variable x_i in the corresponding factor graph.

2. If the subset of variables, $\boldsymbol{x}_s = \{x_i, x_j, x_k, ..\}$ has a common factor node $f_s(\boldsymbol{x}_s)$, then the (unnormalized) joint posterior distribution of this subset of variables conditioned on the set of observed variable is the product of the common factor function $f_s(\boldsymbol{x}_s)$ and all the incoming messages at factor node $f(x_s)$.

Suppose the goal is to determine the marginal posterior of a variable x_j and assume the underlying factor graph is a poly-tree. Then in order to start the

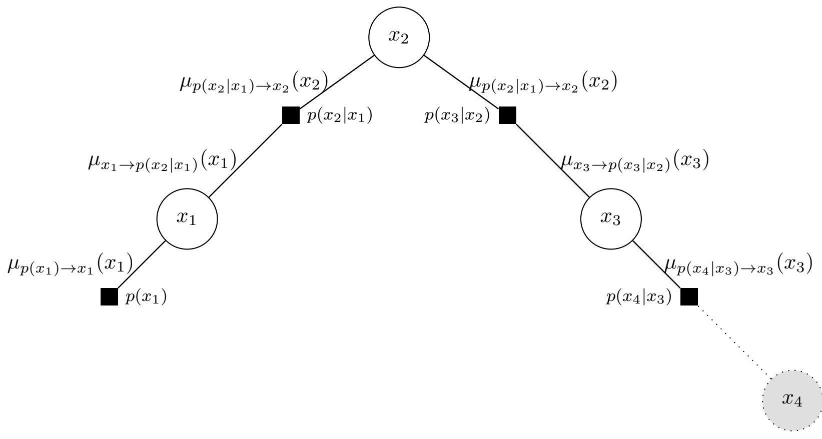

x_2
$\mu_{p(x_2|x_1)\to x_2}(x_2)$
$\mu_{p(x_2|x_1)\to x_2}(x_2)$
$p(x_2|x_1)$
$p(x_3|x_2)$
$\mu_{x_1\to p(x_2|x_1)}(x_1)$
$\mu_{x_3\to p(x_3|x_2)}(x_3)$
x_1
x_3
$\mu_{p(x_1)\to x_1}(x_1)$
$\mu_{p(x_4|x_3)\to x_3}(x_3)$
$p(x_1)$
$p(x_4|x_3)$
x_4

Figure 2.3: Factor graph for the Markov model with x_2 as designated root. The variable x_4 is an observed variable and can therefore be absorbed into the factor $p(x_4|x_3)$.

recursion, the variable node x_j can be interpreted as the root of tree. Then starting from the leafs of the tree, the messages are propagated link by link until the root node is reached from all leafs. Using this scheme ensures that all necessary messages are available for computing the product of the incoming message at node x_j.

The following example illustrates the use of the sum-product algorithm for inference in a simple Markov chain model. Consider the one dimensional discrete time series $\{x_t\}_{t=1}^4$ of length $T = 4$. By assuming the time series is generated by a first order Markov chain [PK11], the joint probability of the time series can be decomposed using the Markov property:

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3) \tag{2.12}$$

where $p(x_1)$ is the initial probability distribution and $p(x_t|x_{t-1})$ are the so-called one-step transition probabilities. Notice, that this distribution is decomposed into 4 local factors.

Consider now the problem of computing the posterior distribution of x_2 conditioned on x_4. The first step is to construct the corresponding factor graph with x_2 as designated root. The resulting factor graph is shown in figure 2.3, where it is seen that the underlying factor graph is indeed a tree. Note that the observed variable x_4 is marked by a shaded circle, which is customary for observed variables. Furthermore, since $x_4 = \hat{x}_4$ is observed it can be considered as a fixed variable and therefore be absorbed into the factor $p(\hat{x}_4|x_3)$, which then becomes a function of x_3 only. For this reason, it is not necessary to include the observed variables in factor graphs. However, in this thesis the observed variable are shown anyway for completeness.

Following the sum-product algorithm, the factor nodes $p(x_1)$ and $p(\hat{x}_4|x_3)$ are identified as leaf nodes. The messages can now be propagated from the leaf nodes to the root. Starting from the right leaf, the message from the leaf node $p(\hat{x}_4|x_3)$ to variable node x_3 is then obtained using eq. (2.11):

$$\mu_{p(\hat{x}_4|x_3)\to x_3}(x_3) = p(\hat{x}_4|x_3) \tag{2.13}$$

and the message from variable node x_3 to factor node $p(x_3|x_2)$ is obtained using eq. (2.8):

$$\mu_{x_3\to p(x_3|x_2)}(x_3) = \mu_{p(\hat{x}_4|x_3)\to x_3}(x_3) = p(\hat{x}_4|x_3) \tag{2.14}$$

Using definition (2.10), the message from factor node $p(x_3|x_2)$ to variable node x_2 is given by the product of the factor function $p(x_3|x_2)$ and the incoming messages except the message from x_2 and then summed over x_3:

$$\mu_{p(x_3|x_2)\to x_2}(x_2) = \sum_{x_3} p(x_3|x_2)\mu_{x_3\to p(x_3|x_2)}(x_3) \tag{2.15}$$

Thus, the root is reached from the right leaf node. Similarly, starting from the left leaf results in the following sequence of messages:

$$\mu_{p(x_1)\to x_1}(x_1) = p(x_1) \tag{2.16}$$
$$\mu_{x_1\to p(x_2|x_1)}(x_1) = \mu_{p(x_1)\to x_1}(x_1) = p(x_1) \tag{2.17}$$
$$\mu_{p(x_2|x_1)\to x_2}(x_2) = \sum_{x_1} p(x_2|x_1)\mu_{x_1\to p(x_2|x_1)}(x_1) \tag{2.18}$$

Finally, using the result in table 2.1 the marginal posterior is obtained as the product of the incoming messages at node x_2:

$$p(x_2|\hat{x}_4) \propto \mu_{p(x_2|x_1)\to x_2}(x_2)\mu_{p(x_3|x_2)\to x_2}(x_2) \tag{2.19}$$

To verify this, the messages can be back-substituted into the above expression:

$$\begin{aligned} p(x_2|\hat{x}_4) &\propto \mu_{p(x_2|x_1)\to x_2}(x_2)\mu_{p(x_3|x_2)\to x_2}(x_2) \\ &= \sum_{x_1} p(x_2|x_1)\mu_{x_1\to p(x_2|x_1)}(x_1) \sum_{x_3} p(x_3|x_2)\mu_{x_3\to p(x_3|x_2)}(x_3) \\ &= \sum_{x_1} p(x_2|x_1)p(x_1) \sum_{x_3} p(x_3|x_2)p(\hat{x}_4|x_3) \\ &= p(\hat{x}_4|x_2)p(x_2) \end{aligned} \tag{2.20}$$

which is seen to be proportional to the true posterior distribution. The normalization constant is easily obtained by summing over x_2. Now suppose the problem is to obtain the marginal posterior distribution of all the latent variable. In principle, the above procedure could be repeated for x_1 and x_3 as root nodes, but then the same quantities would be recomputed multiple times. Instead, x_2 is kept as the root node, but after propagating messages from the leaf to the root, the messages are propagated from the root, i.e. variable node x_2, and back to the leafs. Then all messages in both direction are available and any marginal posterior $p(x_j)$ is simply obtained as the product of the incoming messages at node x_j. That is, by selecting an arbitrary node as root node and propagating all the messages from all leaf nodes to the root node and back to the leaves again, we can obtain all the marginal posterior distributions in the distribution in a very efficient manner [Bis06]. One very interesting aspect of the sum-product algorithm is that a number of seemingly unrelated algorithms can be interpreted as instances of the sum-product algorithm. For example, both Kalman filtering [Bis06] and the forward/backward algorithm for Hidden Markov Models [RJ86] can be shown to be instances of the sum-product algorithm [KFL01]. In fact, the underlying factor graphs share the same topology. Even more surprising, algorithm like Expectation-Maximization and the Fast Fourier Transform can also be seen interpreted as instances of the sum-product algorithm [KFL01, LDH+07].

As stated earlier, the sum-product message passing scheme is only guaranteed to produce exact inference if the underlying factor graph is a poly-tree. But since all messages are local, the exact same message passing scheme can be applied in an iterative manner to cyclic factor graph. The *loopy* message passing scheme is not guaranteed to produce exact inference nor converge, but empirical evidence suggests that when it does converge, it produces accurate estimates of the true posteriors [MWJ99]. In fact, the very successful procedure for decoding the so-called *turbo codes* can be interpreted as an instance of the sum-product algorithm operating in a loopy graph [MMC09].

The Max-Sum Algorithm

Completely analogous to the sum-product algorithm, the max-product algorithm is designed to compute the most likely variable configuration w.r.t. a given probability distribution [Bis06]. That is,

$$\boldsymbol{x}^{max} = \max_{\boldsymbol{x}} p(x_1, x_2, .., x_n) \tag{2.21}$$

Loosely speaking, this algorithm simply correspond to exchanging the summation operators in the sum-product algorithm with maximization operators. However, products of probabilities can result in very small numbers and since computers only have finite accuracy, this can lead to numerical underflow. To avoid this issue, the max-product algorithm can be applied to the logarithm of the probability distribution of interest. Since the logarithm is a monotonically increasing function, the order of the maximization operator and the logarithm can be exchanged, i.e. $\ln \max_x p(x) = \max_x \ln p(x)$. Furthermore, when the logarithm is applied to the product decomposition of the joint probability, it turns into a sum decomposition. The resulting algorithms is therefore referred to as the *max-sum* algorithm.

For non-leafs, the two types of messages then become:

$$\mu_{f_a \to x_i}(x_i) = \max_{x_1, x_2, \dots} \left[\ln f(x_1, x_2, \dots) + \sum_{j \neq i} \mu_{x_j \to f_a} \right] \tag{2.22}$$

$$\mu_{x_i \to f_a}(x_i) = \sum_{b \neq a} \mu_{f_b \to x_i}(x_i) \tag{2.23}$$

while for leaf nodes:

$$\mu_{f_a \to x_i}(x_i) = \ln f(x_i) \tag{2.24}$$

$$\mu_{x_i \to f_a}(x_i) = 0 \tag{2.25}$$

Using these messages, the exact same schedule as for the sum-product algorithm can be used for the max-sum algorithm. However, the above approach would simply return the probability of the most likely configuration of variables and not the most likely configuration of variables itself. In order to obtain this configuration of variables, it is necessary to keep track of the arguments, which maximize each message during the propagation of the messages. Then it is possible to *backtrack* each argument in a *dynamic programming* manner and then obtain the desired configuration [Bis06]. However, we will not go into details with this approach since it is not needed in this thesis.

This ends the introduction to message passing and factors graph and we are now equipped with the necessary tools for deriving the framework of *approximate message passing*.

2.2 Approximate Message Passing

Now the attention is turned towards the *approximate message passing* (AMP) algorithm introduced by Donoho et al. in [DMM09, DMM10]. AMP is a message passing-based framework developed for solving the basis pursuit or the basis pursuit de-noising problem in the context of compressed sensing [EK12]. AMP can also been seen as an instance of the class of iterative thresholding algorithms (see literature review in section 1.2). However, AMP distinguishes itself from the other algorithms in this class by having a significant better sparsity-undersampling trade-off. In fact, Donoho et al. shows that in the high dimensional limit $n, m \to \infty, \quad m/n \to \delta$ the phase transition of AMP approaches that achievable by ℓ_1-minimization, while maintaining low computational complexity [DMM09].

Consider the linear system of equations $\boldsymbol{Ax} = \boldsymbol{y}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{x}_0 \in \mathbb{R}^n$ is the true solution. It is assumed that the columns of $\boldsymbol{A}$ have been scaled to unit ℓ_2-norm. In the remainder of this thesis, the undersamplingsratio δ of a given problem will be defined as $\delta = m/n$, k will denote the number of non-zero elements in the true solution and the sparsity will be defined as $\rho = k/m$. For the basis pursuit problem, the simple update equations for AMP are then given by

$$\boldsymbol{x}^{k+1} = \eta\left(\boldsymbol{A}^T\boldsymbol{z}^k + \boldsymbol{x}^t;\, \hat{\tau}^k\right), \tag{2.26}$$

$$\boldsymbol{z}^k = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^k + \frac{1}{\delta}\boldsymbol{z}^{t-1}\left\langle \eta'\left(\boldsymbol{A}^T\boldsymbol{z}^{t-1} + \boldsymbol{x}^{t-1};\, \hat{\tau}^{t-1}\right)\right\rangle, \tag{2.27}$$

$$\hat{\tau}^k = \frac{\hat{\tau}^{t-1}}{\delta}\left\langle \eta'\left(\boldsymbol{A}^T\boldsymbol{z}^{t-1} + \boldsymbol{x}^k;\, \hat{\tau}^{t-1}\right)\right\rangle, \tag{2.28}$$

where $\eta(x, \tau)$ is a soft thresholding function applied component-wise, $\eta'(x, \tau)$ is its derivative, $\langle\cdot\rangle$ is the averaging operator and k is the iteration index. Due to the form of the update equations, it is readily seen that this algorithm belongs to the class of iterative thresholding algorithms.

Analogously, the update equations for the basis pursuit de-noising (BPDN) problem, i.e. $\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{e}$, are very similar:

$$\boldsymbol{x}^{k+1} = \eta\left(\boldsymbol{A}^T\boldsymbol{z}^k + \boldsymbol{x}^k;\, \lambda + \gamma^k\right), \tag{2.29}$$

$$\boldsymbol{z}^k = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^k + \frac{1}{\delta}\boldsymbol{z}^{t-1}\left\langle \eta'\left(\boldsymbol{A}^T\boldsymbol{z}^{t-1} + \boldsymbol{x}^{t-1};\, \lambda + \gamma^{t-1}\right)\right\rangle, \tag{2.30}$$

$$\gamma^k = \frac{\lambda + \gamma^{t-1}}{\delta}\left\langle \eta'\left(\boldsymbol{A}^T\boldsymbol{z}^{t-1} + \boldsymbol{x}^k;\, \lambda + \gamma^{t-1}\right)\right\rangle, \tag{2.31}$$

where λ is the regularization parameter. By comparing the update equations for BP and BPDN, it is seen that the two algorithms are identical for $\lambda = 0$ and therefore we will only focus on the latter without loss of generality.

Derivation of AMP

The purpose of this subsection is to the describe the derivation of AMP following the approach in [DMM10]. Since the derivation is rather lengthy, it is divided into 4 parts:

- Part 1: Derive exact update rules using the sum-product algorithm
- Part 2: Taking the large system limit
- Part 3: Taking the large β limit
- Part 4: Reducing the number of messages

We will now dive directly into the first part.

Part 1: Derive Update Rules using the Sum-Product Algorithm

First the underlying linear model is defined. It is assumed that the prior distribution over each component of the solution is a Laplace distribution with a common hyperparameter β:

$$p\left(x_i\right) = \frac{\beta\lambda}{2}\exp\left(-\beta\lambda\left|x_i\right|\right), \qquad \beta \geq 0 \tag{2.32}$$

Similarly, the noise is assumed to be independent and Gaussian distributed, i.e., the likelihood function is given by:

$$p\left(\boldsymbol{y}|\boldsymbol{x}\right) = \mathcal{N}\left(\boldsymbol{y}\,\middle|\,\boldsymbol{A}\boldsymbol{x}, \beta^{-1}\boldsymbol{I}_m\right) \tag{2.33}$$

where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix and β is the precision of the noise. Note, that in this model the prior distribution and the likelihood share the hyperparameter β. This gives rise to the following joint distribution:

$$\begin{aligned} p\left(\boldsymbol{x}, \boldsymbol{y}\right) &= p\left(\boldsymbol{y}|\boldsymbol{x}\right) p\left(\boldsymbol{x}\right) \\ &= \prod_{a=1}^{m} p\left(y_a|\boldsymbol{x}\right) \prod_{i=1}^{n} p\left(x_i\right) \end{aligned} \tag{2.34}$$

Now the corresponding factor graph is set up, where $\boldsymbol{x}$ is considered a latent variables and $\boldsymbol{y}$ an observed variable. The resulting factor graph is shown in figure 2.4.

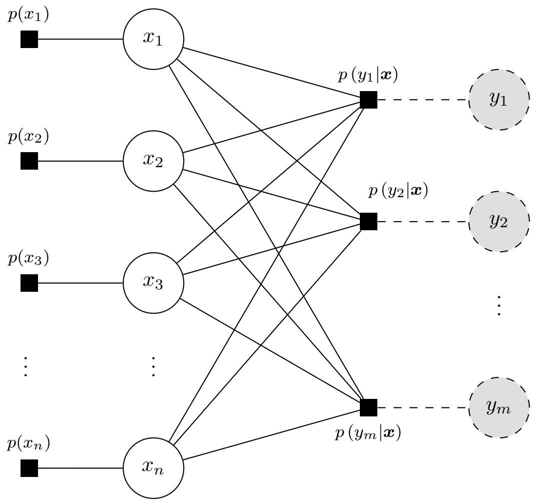


Figure 2.4: Factor graph for the joint distribution given in eq. (2.34)

Inspecting the figure reveals that the factor graph contains multiple loops and therefore it is necessary to resort to loopy message passing. The next step is then to derive the loopy sum-product messages for the posterior of x_i.

In remainder of this thesis, $[m]$ will be used to denote the set of indices from 1 to m, i.e. $[m] = \{a \middle| a \in \mathbb{N},\, 1 \leq a \leq m\}$. Furthermore, the variables $i, j \in [n]$ will be used as indices for variable nodes and the variables $a, b \in [m]$ will be used as indices for the factor nodes.

The sum-product algorithm states that the posterior marginal distribution over x_i conditioned on $\boldsymbol{y}$ is given by the product of the incoming messages at variable node x_i:

$$p\left(x_i|\boldsymbol{y}\right) = \mu_{p(x_i)\rightarrow x_i}(x_i) \prod_a \mu_{p(y_a|\boldsymbol{x})\rightarrow x_i}(x_i) \tag{2.35}$$

The sum-product rules decribed in section 2.1 are now used to derive the messages based on this factor graph. The message from $p(x_i)$ to x_i is trivial, since $p(x_i)$ is a leaf node

$$\mu_{p(x_i)\rightarrow x_i}(x_i) = p_i(x_i) = \frac{\beta\lambda}{2} \exp\left(-\beta\lambda\left|x_i\right|\right) \tag{2.36}$$

That is, the message simply corresponds to the prior density. Next, the message from factor node $p(y_a \big| \boldsymbol{x})$ to variable node x_i is given by

$$\begin{aligned}
\mu_{p(y_a|\boldsymbol{x})\to x_i}(x_i) &= \int p(y_a \big| \boldsymbol{x}) \prod_{j\neq i} \mu_{x_j \to p(y_a|\boldsymbol{x})}(x_j)\, \mathrm{d}x_{j\neq i} \\
&= \int \frac{1}{\sqrt{2\pi\beta^{-1}}} \exp\left[-\frac{\beta}{2}\left(y_a - (\boldsymbol{A}\boldsymbol{x})_a\right)^2\right] \prod_{j\neq i} \mu_{x_j \to p(y_a|\boldsymbol{x})}(x_j)\, \mathrm{d}x_{j\neq i} \\
&\propto \int \exp\left[-\frac{\beta}{2}\left(y_a - (\boldsymbol{A}\boldsymbol{x})_a\right)^2\right] \prod_{j\neq i} \mu_{x_j \to p(y_a|\boldsymbol{x})}(x_j)\, \mathrm{d}x_{j\neq i} \qquad (2.37)
\end{aligned}$$

where $\propto$ means equal up to a constant factor and the notation $\mathrm{d}x_{j\neq i}$ means integration over the set of variables $\left\{x_j \big| j \neq i\right\}$. Finally, the message from variable node x_i to factor node $p(y_a \big| \boldsymbol{x})$ is then given by the product of incoming messages at node x_i, except the message from $p(y_a \big| \boldsymbol{x})$ itself:

$$\begin{aligned}
\mu_{x_i \to p(y_a|\boldsymbol{x})}(x_i) &= \mu_{p(x_i)\to x_i}(x_i) \prod_{b\neq a} \mu_{p(y_b|\boldsymbol{x})\to x_i}(x_i) \\
&= \frac{\beta\lambda}{2} \exp\left(-\beta\lambda \left|x_i\right|\right) \prod_{b\neq a} \mu_{p(y_b|\boldsymbol{x})\to x_i}(x_i) \\
&\propto \exp\left(-\beta\lambda \left|x_i\right|\right) \prod_{b\neq a} \mu_{p(y_b|\boldsymbol{x})\to x_i}(x_i) \qquad (2.38)
\end{aligned}$$

Inspection of the messages from variables x_i to factors $p(y_a|\boldsymbol{x})$ shows that in order to compute $\mu_{x_i\to p(y_a|\boldsymbol{x})}(x_i)$, the message $\mu_{p(y_b|\boldsymbol{x})\to x_i}(x_i)$ is needed and vice versa. This is a manifestation of the problem of message passing in loopy factor graphs. However, resorting to loopy message passing yields the following update scheme

$$\mu^{k+1}_{x_i\to g_a}(x_i) \propto \exp\left(-\beta\lambda \left|x_i\right|\right) \prod_{b\neq a} \mu^{k}_{g_b\to x_i}(x_i) \qquad (2.39)$$

$$\mu^{k}_{g_a\to x_i}(x_i) \propto \int \exp\left[-\frac{\beta}{2}\left(y_a - (\boldsymbol{A}\boldsymbol{x})_a\right)^2\right] \prod_{j\neq i} \mu^{k}_{x_j\to g_a}(x_j)\, \mathrm{d}x_{j\neq i} \qquad (2.40)$$

where superscript k denotes iteration number. To ease the notation, the following notation will be adopted in the remainder of the AMP derivation:

$$\begin{aligned}
\mu^{k}_{x_i\to p(y_a|\boldsymbol{x})}(x_i) &= \mu^{k}_{i\to a}(x_i) \\
\mu^{k}_{p(y_a|\boldsymbol{x})\to x_i}(x_i) &= \mu^{k}_{a\to i}(x_i) \qquad (2.41)
\end{aligned}$$

The goal is now to approximate the above message passing scheme.

Part 2: Taking the Large System Limit

It is now justified that the both types of messages can be well approximated by simple parametric densities in the large system limit, i.e. when $m, n \to \infty$ for $m/n \to \delta$. In particular, in this limit the messages from factor nodes to variable nodes, $\mu_{a\to i}(x_i)$, are approximated by a Gaussian density and the messages from variable nodes to factor nodes are approximated by the product of a Laplace density and a Gaussian density. Of course, linear systems with an infinite number of equations and infinite number of unknowns are purely mathematical constructions. But for many linear systems of interest, the number of unknowns is huge, e.g. the number of unknowns in imaging applications can be of order of 10^6 [RS08].

The messages from the factor nodes to the variable nodes, i.e $\mu_{a\to x_i}(x_i)$, are first considered. Using a variant of the Berry-Esseen[2] central limit theorem [Dur04, DMM10], it is possible to show that in the limit $n \to \infty$, the messages $\mu_{a\to i}(x_i)$ converge to a Gaussian distribution w.r.t. supremum norm. To show this, define the mean and variance of a random variable distributed according to density $p(x_i) \propto \mu_{i\to a}(x_i)$ as $x^k_{i\to a}$ and $\frac{1}{\beta}\tau^k_{i\to a}$, respectively:

$$x^k_{i\to a} \equiv \mathbb{E}\left[x_i\right], \quad \text{for} \quad x_i \sim \mu_{x_i\to a}(x_i) \tag{2.42}$$

$$\frac{1}{\beta}\tau^k_{i\to a} \equiv \mathbb{V}\left[x_i\right] \tag{2.43}$$

Next, notice that eq. (2.40) can be written as an expectation over the messages $\prod_{j\neq i}\mu^k_{j\to a}(x_j)$ and rewritten as:

$$\begin{aligned}\mu^k_{a\to x_i}(x_i) &\propto \mathbb{E}\left[\exp\left(-\frac{\beta}{2}\left(y_a - (\boldsymbol{Ax})_a\right)^2\right)\right]\\ &= \mathbb{E}\left[\exp\left(-\frac{\beta}{2}\left(y_a - A_{ai}x_i - \sum_{j\neq i}A_{aj}x_j\right)^2\right)\right],\end{aligned} \tag{2.44}$$

where it is used that $(\boldsymbol{Ax})_a = \sum_i A_{ai}x_i = A_{ai}x_i + \sum_{j\neq i}A_{aj}x_j$. Now define an auxiliary variable Z and an auxiliary function $h_{x_i}(z)$ as:

$$Z \equiv y_a - \sum_{j\neq i}A_{aj}x_j, \qquad h_{x_i}(z) \equiv \exp\left[\frac{\beta}{2}\left(A_{ai}x_i - z\right)^2\right] \tag{2.45}$$

[2]The Berry-Esseen theorem is a variant of the central limit theorem, which quantifies the rate at which a sum of random variables converge to a Gaussian density. This particular variant is proved in the appendix in [DMM10].

The mean and variance of Z are now easily computed using the linearity properties of the expectation operator. Denote the mean and variance as $z^k_{a\to i}$ and $\hat{\tau}^k_{a\to i}$, respectively:

$$z^k_{a\to i} \equiv \mathbb{E}\left[Z\right] = \mathbb{E}\left[y_a\right] - \sum_{j\neq i} A_{aj}\mathbb{E}\left[x_j\right] = y_a - \sum_{j\neq i} A_{aj} x^k_{j\to a} \tag{2.46}$$

$$\hat{\tau}^k_{a\to i} \equiv \mathbb{V}\left[Z\right] = \mathbb{V}\left[y_a\right] + \sum_{j\neq i} A^2_{aj}\mathbb{V}\left[x_j\right] = \frac{1}{\beta}\sum_{j\neq i} A^2_{aj}\tau^k_{j\to a} \tag{2.47}$$

Substituting the two definitions in eq. (2.45) into eq. (2.44) yields

$$\mu^k_{a\to x_i}(x_i) \propto \mathbb{E}\left[h_{x_i}(Z)\right] \tag{2.48}$$

Now let W be a Gaussian distributed random variable with mean $z^k_{a\to t}$ and variance $\frac{1}{\beta}\hat{\tau}^k_{a\to i}$, i.e. the same mean and variance as Z, then the Berry-Esseen central limit theorem (see appendix A.2) implies:

$$\sup_{x_i\in\mathbb{R}} \left|\mu^k_{a\to i}(x_i) - \mathbb{E}\left[h_{x_i}(W)\right]\right| \leq \frac{C_k}{n^{\frac{1}{2}}\left(\hat{\tau}^k_{a\to i}\right)^{\frac{3}{2}}} \tag{2.49}$$

where C_k is a constant. This implies that the messages $\mu^k_{a\to x_i}(x_i)$ will converge to the function $\mathbb{E}\left[h_{x_i}(W)\right]$ as n approach infinity. Thus, to show that the messages $\mu^k_{a\to x_i}(x_i)$ converge to a Gaussian density, it is necessary to show that

$$\frac{\mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i} \tag{2.50}$$

corresponds to a Gaussian density.

Recall that $W \sim \mathcal{N}\left(z^k_{a\to t}, \frac{1}{\beta}\hat{\tau}^k_{a\to i}\right)$ and consider the numerator of the above expression:

$$\begin{aligned}\mathbb{E}\left[h_{x_i}(W)\right] &= \int h_{x_i}(W)\mathcal{N}\left(W|z^k_{a\to i}, \frac{\hat{\tau}^k_{a\to}i}{\beta}\right)\mathrm{d}W\\ &= \int \exp\left[\frac{\beta}{2}\left(A_{ai}x - W\right)^2\right]\mathcal{N}\left(W|z^k_{a\to i}, \frac{\hat{\tau}^k_{a\to}i}{\beta}\right)\mathrm{d}W\end{aligned}$$

It is now used that the first factor corresponds to an unnormalized Gaussian density and followed by an application of Gaussian multiplication rule (see appendix A.1), we get:

$$\begin{aligned}
\mathbb{E}\left[h_{x_i}(W)\right] &= \int \sqrt{\frac{2\pi}{\beta}} \mathcal{N}\left(W|A_{ai}x_i, \frac{1}{\beta}\right) \mathcal{N}\left(W|z^k_{a\to i}, \frac{\hat{\tau}^k_{a\to} i}{\beta}\right) \mathrm{d}W \\
&= \sqrt{\frac{2\pi}{\beta}} \int \mathcal{N}\left(W|\frac{A_{ai}x_i\beta + \frac{z^k_{a\to i}\beta}{\hat{\tau}^k_{a\to i}}}{\beta + \frac{\beta}{\hat{\tau}^k_{a\to i}}}, \frac{1}{\beta + \frac{1}{\hat{\tau}^k_{a\to i}}}\right) \mathcal{N}\left(0|A_{ai}x_i - z^k_{a\to i}, \frac{1}{\beta} + \frac{\hat{\tau}^k_{a\to i}}{\beta}\right) \mathrm{d}W \\
&= \sqrt{\frac{2\pi}{\beta}} \mathcal{N}\left(0|A_{ai}x_i - z^k_{a\to i}, \frac{1}{\beta} + \frac{\hat{\tau}^k_{a\to i}}{\beta}\right) \int \mathcal{N}\left(W|\frac{A_{ai}x_i\beta + \frac{z^k_{a\to i}\beta}{\hat{\tau}^k_{a\to i}}}{\beta + \frac{\beta}{\hat{\tau}^k_{a\to i}}}, \frac{1}{\beta + \frac{1}{\hat{\tau}^k_{a\to i}}}\right) \mathrm{d}W,
\end{aligned}$$

This is simplified using the fact that probability densities integrate to 1:

$$\begin{aligned}
\mathbb{E}\left[h_{x_i}(W)\right] &= \sqrt{\frac{2\pi}{\beta}} \mathcal{N}\left(0|A_{ai}x_i - z^k_{a\to i}, \frac{1}{\beta} + \frac{\hat{\tau}^k_{a\to i}}{\beta}\right) \\
&= \sqrt{\frac{2\pi}{\beta}} \mathcal{N}\left(A_{ai}x_i \middle| z^k_{a\to i}, \frac{1}{\beta} + \frac{\hat{\tau}^k_{a\to i}}{\beta}\right) \qquad (2.51)
\end{aligned}$$

Using this results, the integral in denominator of eq. (2.50) is easily computed by changing variable of integration to $\hat{x}_i = A_{ai}x_i$. This yields:

$$\int \mathbb{E}\left[h_{x_i}(W)\right] \mathrm{d}x_i = \frac{1}{A_{ai}} \sqrt{\frac{2\pi}{\beta}} \qquad (2.52)$$

Now the expression in eq. (2.50) is easily computed by combining eq. (2.51) and eq. (2.52):

$$\begin{aligned}
\frac{\mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right] \mathrm{d}x_i} &= \frac{\sqrt{\frac{2\pi}{\beta}} \mathcal{N}\left(A_{ai}x_i | z^k_{a\to i}, \frac{1}{\beta} + \frac{\hat{\tau}^k_{a\to i}}{\beta}\right)}{\frac{1}{A_{ai}} \sqrt{\frac{2\pi}{\beta}}} \\
&= A_{ai} \mathcal{N}\left(A_{ai}x_i \middle| z^k_{a\to i}, \frac{1 + \hat{\tau}^k_{a\to i}}{\beta}\right) \qquad (2.53)
\end{aligned}$$

Note, that the factor A_{ai} appears outside, because the density is defined over $A_{ai}x_i$ and not x_i alone. Finally, by combining the inequality in eq. (2.49) with the above in eq. (2.53), we conclude:

$$\mu^k_{a\to i}(x_i) \propto \mathcal{N}\left(A_{ai}x_i \middle| z^k_{a\to i}, \frac{1 + \hat{\tau}^k_{a\to i}}{\beta}\right) \qquad \text{for} \qquad n \to \infty \qquad (2.54)$$

Thus, in the large system limit, the messages from factor nodes to variable nodes simplifies to Gaussian densities parametrized by $z^k_{a\to i}$ and $\hat{\tau}^k_{a\to i}$.

Next, consider the messages from variable nodes to factor nodes, i.e. $\mu_{i\to a}(x_i)$. For that purpose, define the family of densities $f_\beta(x; s, b)$ by:

$$f_\beta(x; s, b) \equiv \frac{1}{Z_\beta} \exp\left[-\beta\left|x\right| - \frac{\beta}{2b}\left(s - x\right)^2\right] \qquad (2.55)$$

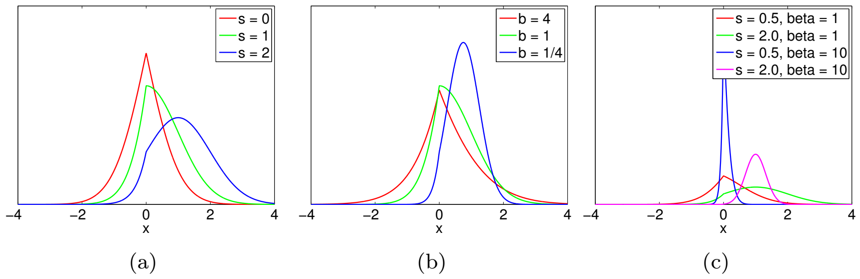


Figure 2.5: Plots of the density f for different values of the parameters, which are shown in the legends. (a) the effects of varying the s-parameter for $b = \beta = 1$. (b) the effect of varying β. It is seen that for large β's the distribution becomes more "spiky"

This particular family of densities are recognized as a product of a Laplace density and a Gaussian density and the figures 2.5(a)-(c) show the form of this density for different parameter combinations. The idea is now to show that the messages $\mu_{i\to a}(x_i)$ can be approximated by this density. Denote the mean and variance of the density $f_\beta(x; s, b)$ as $\mathsf{F}_\beta(s, b)$ and $\mathsf{G}_\beta(s, b)$, respectively:

$$\mathsf{F}_\beta(s,b) = \mathbb{E}\left[X\right], \quad \text{where} \quad X \sim f_\beta(x; s, b) \tag{2.56}$$

$$\mathsf{G}_\beta(s,b) = \mathbb{V}\left[X\right] \tag{2.57}$$

Recall from eq. (2.39), that the messages from variable node to factor nodes are given by

$$\mu^{k+1}_{x_i\to g_a}(x_i) \propto \exp\left(-\beta\lambda\left|x_i\right|\right)\prod_{b\neq a}\mu^k_{b\to i}(x_i)$$

Plugging in the result from eq. (2.53) and simplifying yields:

$$\begin{aligned}\mu^{k+1}_{i\to a}(x_i) &\propto \exp\left(-\beta\lambda\left|x_i\right|\right)\prod_{b\neq a}A_{bi}\mathcal{N}\left(A_{bi}x_i\middle|z^k_{b\to i}, \frac{1+\hat{\tau}^k_{b\to i}}{\beta}\right)\\ &= \exp\left(-\beta\lambda\left|x_i\right|\right)\prod_{b\neq a}\frac{A_{bi}}{\sqrt{2\pi\frac{1+\hat{\tau}^k_{b\to i}}{\beta}}}\exp\left[-\frac{\beta}{2\left(1+\hat{\tau}^k_{a\to i}\right)}\left(A_{bi}x_i - z^k_{b\to i}\right)^2\right]\\ &\propto \exp\left[-\beta\lambda\left|x_i\right| - \sum_{b\neq a}\frac{\beta}{2\left(1+\hat{\tau}^k_{a\to i}\right)}\left(A_{bi}x_i - z^k_{b\to i}\right)^2\right] \end{aligned}\tag{2.58}$$

The parameters $\hat{\tau}^k_{a\to i}$, which are defined in eq. (2.47), do only depend on index i through the "missing term in the sum". Moreover, the columns of $\boldsymbol{A}$ are assumed to have unit ℓ_2-norm, and therefore the elements A_{ai}^2 are expected to be very small for a large values of m. Hence, parameters $\hat{\tau}_{a\to i}^k$ can be approximated by a common parameter $\hat{\tau}^k$ and this approximation is expected to become negligible in the large system limit:

$$\hat{\tau}^k = \hat{\tau}_{a\to i}^k \tag{2.59}$$

Substituting this approximation into eq. (2.58):

$$\mu_{i\to a}^{k+1}(x_i) \propto \exp\left[-\beta\lambda\,|x_i| - \frac{\beta}{2\left(1+\hat{\tau}^k\right)}\sum_{b\neq a}\left(A_{bi}x_i - z_{b\to i}^k\right)^2\right]$$

Expanding the term in the parenthesis and rearranging yields

$$\mu_{i\to a}^{k+1}(x_i) \propto \exp\left[-\beta\lambda\,|x_i| - \frac{\beta}{2\left(1+\hat{\tau}^k\right)}\left(x_i^2\sum_{b\neq a}A_{bi}^2 + \sum_{b\neq a}\left(z_{b\to i}^k\right)^2 - 2x_i\sum_{b\neq a}A_{bi}z_{b\to i}^k\right)\right]$$

Since $z_{b\to i}^k$, as defined in eq. (2.46), is constant w.r.t. x_i, the second term of the exponent can be "absorbed" by the normalization constant. The fact that the columns of $\boldsymbol{A}$ have unit ℓ_2-norm implies that $\sum_{b\neq a}A_{bi}^2 \approx 1 - \frac{1}{m}$. Inserting this approximation yields:

$$\mu_{i\to a}^{k+1}(x_i) \propto \exp\left[-\beta\lambda\,|x_i| - \frac{\beta}{2\left(1+\hat{\tau}^k\right)}\left(x_i^2 - \frac{x_i^2}{m} - 2x_i\sum_{b\neq a}A_{bi}z_{b\to i}^k\right)\right]$$

When m is sufficiently large, the term $\frac{x_i^2}{m}$ becomes negligible. Ignoring the term $\frac{x_i^2}{m}$ and completing the square over x_i again yields:

$$\mu_{i\to a}^{k+1}(x_i) \propto \exp\left[-\beta\lambda\,|x_i| - \frac{\beta}{2\left(1+\hat{\tau}^k\right)}\left(x_i - \sum_{b\neq a}A_{bi}z_{b\to i}^k\right)^2\right]$$

The hyperparameter λ is positive by definition and can therefore be moved inside the absolute value operator. Furthermore, by multiplying and dividing by λ^2 in the last term, we obtain

$$\mu_{i\to a}^{k+1}(x_i) \propto \exp\left[-\beta\,|\lambda x_i| - \frac{\beta}{2\lambda^2\left(1+\hat{\tau}^k\right)}\left(\lambda x_i - \lambda\sum_{b\neq a}A_{bi}z_{b\to i}^k\right)^2\right]$$

Then by comparing the above result with the family of densities in (2.55), it is clear that:

$$\mu_{i\to a}^{k+1}(x_i) \propto \lambda f_\beta\left(\lambda x_i \,\middle|\, \lambda\sum_{b\neq a}A_{bi}z_{b\to i}^k,\ \lambda^2\left(1+\hat{\tau}^k\right)\right) \tag{2.60}$$

That is, it has now been shown that the messages from variable nodes to factor nodes, i.e. $\mu_{i\to a}(x_i)$, can be described by the density f_β in the large system limit.

Furthermore, using the definition of $x^k_{i\to a}$ in eq. (2.42) and the definition of $\frac{1}{\beta}\tau^k_{i\to a}$ in eq. (2.43), we have that

$$x^{k+1}_{i\to a} = \frac{1}{\lambda}\mathsf{F}_\beta\left(\lambda\sum_{b\neq a}A_{bi}z^k_{b\to i},\ \lambda^2\left(1+\hat{\tau}^k\right)\right) \tag{2.61}$$

$$\hat{\tau}^{k+1}_{i\to a} = \frac{\beta}{\lambda^2}\mathsf{G}_\beta\left(\lambda\sum_{b\neq a}A_{bi}z^k_{b\to i},\ \lambda^2\left(1+\hat{\tau}^k\right)\right) \tag{2.62}$$

where the factors $\frac{1}{\lambda}$ and $\frac{1}{\lambda^2}$ appear because x_i is scaled with λ on the right hand side of eq. (2.60). Using the approximation $\tau^k_{i\to a} = \tau^k$, we get

$$\hat{\tau}^{k+1} = \frac{1}{m}\frac{\beta}{\lambda^2}\sum_{i=1}^{n}\mathsf{G}_\beta\left(\lambda\sum_{b}A_{bi}z^k_{b\to i},\ \lambda^2\left(1+\hat{\tau}^k\right)\right) \tag{2.63}$$

Notice, the summation is over n terms, but we divide by m to take into account that we only have m degrees of freedom.

The two types of messages, i.e. $\mu_{i\to a}(x_i)$ and $\mu_{a\to i}(x_i)$, are now reduced to simple parametric densities and thus, the message passing scheme has now been greatly simplified.

Part 3: Taking the Large β Limit

We will now show that in limit $\beta\to\infty$, the mean of the distribution f_β is described by a soft thresholding function and similarly, the variance is described by the derivative of this soft thresholding function. The integrals, which define the mean and variance of the density f_β, are by definition:

$$\mathsf{F}_\beta\left(s,\ b\right) = \int x\,\frac{1}{Z_\beta}\exp\left[-\beta\left|x\right| - \frac{\beta}{2b}\left(x-s\right)^2\right]\mathrm{d}x \tag{2.64}$$

$$\mathsf{G}_\beta(s,b) = \int\left(x-\mathsf{F}(s,b)\right)^2\frac{1}{Z_\beta}\exp\left(-\beta\left|x\right| - \frac{\beta}{2b}\left(x-s\right)^2\right)\mathrm{d}x \tag{2.65}$$

First consider the expression for the mean value. In the limit $\beta\to\infty$, the exponential function in eq. (2.64) will approach a Dirac's delta function at the value of x, which maximizes the exponent, i.e. x^*. Hence, by the sift property of Dirac's delta functions under integrals, the mean value is simply equal to the value of x^*. That is,

$$\lim_{\beta\to\infty} \mathsf{F}_\beta(s,\ b) = x^* = \arg\max_x \left\{ -|x| - \frac{1}{2b}(x-s)^2 \right\}$$

Next, by analyzing the partial derivative of $-|x| - \frac{1}{2b}(x-s)^2$ w.r.t. x for $x < 0$ and for $0 < x$, it can be shown that x^* is determined by:

$$x^* = \left\{ \begin{array}{ll} s+b, & \text{if } \ -s > b \\ s-b, & \text{if } \quad\ s > b \\ 0, & \text{if } \ |s| < b \end{array} \right\} = \operatorname{sign}(s)\left(|s| - b\right) \tag{2.66}$$

To summarize, in the large β limit, we can write the mean value of the $f_\beta(x; s, b)$-density as:

$$\mathsf{F}(s, b) = x^* = \eta(s, b) \equiv \operatorname{sign}(s)\left(|s| - b\right) \tag{2.67}$$

where $\eta(s, b)$ is referred to as the soft thresholding function.

Next, consider $\mathsf{G}_\beta(s, b)$, i.e. the variance of $f_\beta(x; s, b)$, in the large β limit. Analysing the exponential function in eq. (2.65) in the limit $\beta \to \infty$, shows that the density f_β can be approximated by a Laplace density and a Gaussian density. In the case $|s| < b$, the first term $-\beta|x|$ is the dominating term in the exponent of the f_β-density. Therefore, the resulting density can be approximated by an Laplace distribution with mean $x^* = 0$ and variance $\frac{2}{\beta^2}$. On the contrary, when $|s| \geq b$, the second term in the exponent is dominating and therefore the resulting density can be approximated by a Gaussian distribution with mean $x^* = \operatorname{sign}(s)\left(|s| - b\right)$ and variance $\frac{b}{\beta}$.

From eq. (2.43), we have $\tau^k_{i\to a} = \beta \mathsf{G}_\beta(s, b)$, therefore we consider the limit of $\beta \mathsf{G}_\beta(s, b)$. Therefore,

$$\begin{aligned} \lim_{\beta\to\infty} \beta \mathsf{G}_\beta(s, b) &= \begin{cases} \lim\limits_{\beta\to\infty} \beta\frac{2}{\beta^2} & \text{if } |s| < b \\ \lim\limits_{\beta\to\infty} \beta\frac{b}{\beta} & \text{if } |s| \geq b \end{cases} \\ &= \begin{cases} 0 & \text{if } |s| < b \\ b & \text{if } |s| \geq b \end{cases} \\ &= b \cdot \eta'(s, b) \end{aligned} \tag{2.68}$$

Thus, it is seen that the mean and variance of the $f(x; s, b)$-densities can be

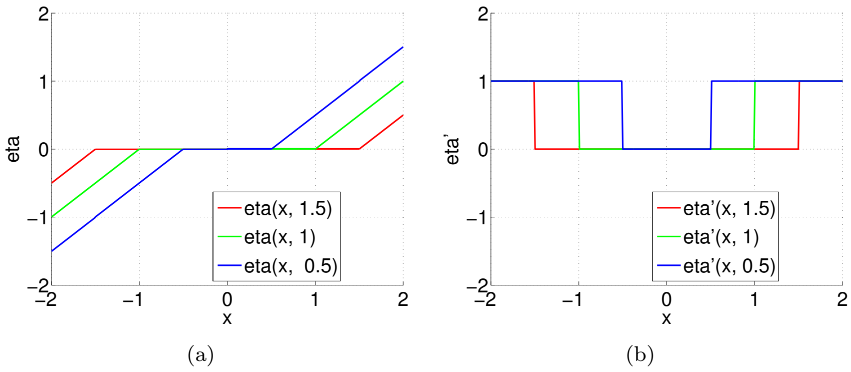


Figure 2.6: (a) The left-most figure shows the function $\eta(x, b)$ for $b = \{0.5, 1, 1.5\}$, where it is seen that $\eta(x, b)$ is equal the soft thresholding function. (b) The right-most figure shows the (piecewise) derivative of $\eta(x, b)$ for $b = \{0.5, 1, 1.5\}$. As seen, it acts like a simply hard thresholding function, i.e. it is zero when the magnitude of the first argument is smaller than b.

written in terms of the soft thresholding function and its (piecewise)[3] derivative. Figure 2.6(a) shows a plot of eq. (2.66) for $b = \{0.5, 1, 1.5\}$, where it is seen that $\eta(x, b)$ indeed act as a soft thresholding function with threshold b.

We can now substitute these explicit expressions for the mean and variance into the update equations. First we substitute the expression for the mean value into eq. (2.61) to get

$$x_{i\to a}^{k+1} = \frac{1}{\lambda}\eta\left(\lambda\sum_{b\neq a} A_{bi} z_{b\to i}^{k},\ \lambda^2\left(1+\hat{\tau}^k\right)\right)$$

Notice, that for $k > 0$, it holds that $k\eta(s, b) = \eta(ks, kb)$. Using this result yields:

$$x_{i\to a}^{k+1} = \eta\left(\sum_{b\neq a} A_{bi} z_{b\to i}^{k},\ \lambda\left(1+\hat{\tau}^k\right)\right)$$

[3] $\eta(\cdot,\cdot)$ is not differentiable at 2 points (the kinks). But according to [DMM09], this does not change the results as long as the $\eta(\cdot,\cdot)$ is Lipschitz continuous, which indeed is the case

Now we substitute the expression for the variance into eq. (2.63) to get

$$\begin{aligned}\hat{\tau}^{k+1} &= \frac{1}{m}\frac{1}{\lambda^2}\sum_{i=1}^{n}\lambda^2\left(1+\hat{\tau}^k\right)\eta'\left(\lambda\sum_b A_{bi}z^k_{b\to i},\ \lambda^2\left(1+\hat{\tau}^k\right)\right)\\ &= \frac{\left(1+\hat{\tau}^k\right)}{m}\sum_{i=1}^{n}\eta'\left(\lambda\sum_b A_{bi}z^k_{b\to i},\ \lambda^2\left(1+\hat{\tau}^k\right)\right)\end{aligned}$$

For $k > 0$, it holds that $\eta'(s,b) = \eta'(ks,kb)$. Therefore,

$$\hat{\tau}^{k+1} = \frac{\left(1+\hat{\tau}^k\right)}{m}\sum_{i=1}^{n}\eta'\left(\sum_b A_{bi}z^k_{b\to i},\ \lambda\left(1+\hat{\tau}^k\right)\right)$$

The message passing scheme has now been simplified to the following update equations:

$$x^{k+1}_{i\to a} = \eta\left(\sum_{b\neq a} A_{bi}z^k_{b\to i},\ \lambda\left(1+\hat{\tau}^k\right)\right), \qquad z^k_{a\to i} = y_a - \sum_{j\neq i} A_{aj}x^k_{j\to a} \tag{2.69}$$

$$\hat{\tau}^{k+1} = \frac{\left(1+\hat{\tau}^k\right)}{m}\sum_{i=1}^{n}\eta'\left(\sum_b A_{bi}z^k_{b\to i},\ \lambda\left(1+\hat{\tau}^k\right)\right) \tag{2.70}$$

Later, this particular message passing scheme is referred to as the MP-scheme (in contrast to AMP-scheme).

In each iterations we have to propagate $2mn$ messages, since we have m messages for each x_i for $i = 1..n$ and n messages for each z_a for $a = 1..m$. Therefore, Donoho et al. [DMM10] introduces yet another approximation, which is described in the following.

Part 4: Reducing the Number of Messages

By analysing the expression for $x^k_{i\to a}$ in eq. (2.69), it is seen that $x^k_{i\to a}$ only depends weakly on index a, since the right hand side only depends on index a through the "missing" term in the sum. Analogously, the same is true for $z^k_{a\to i}$ and index i. The idea is therefore to assume $x^k_{i\to a}$ and $z^k_{a\to i}$ can be approximated as follows:

$$x^k_{i\to a} = x^k_i + \epsilon\cdot x^k_{i\to a} + \mathcal{O}(1/m) \tag{2.71}$$

$$z^k_{a\to i} = z^k_a + \epsilon\cdot z^k_{a\to i} + \mathcal{O}(1/m) \tag{2.72}$$

where $\epsilon << 1$ is a positive small number. Neglecting the $\mathcal{O}(1/m)$ terms and substituting these approximations into the expressions in eq. (2.69) leads to:

$$\begin{aligned} x_i^k + \epsilon \cdot x_{i\to a}^k &= \eta\left(\sum_{b\neq a} A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right), \lambda\left(1+\hat{\tau}^k\right)\right) \\ z_a^k + \epsilon \cdot z_{a\to i}^k &= y_a - \sum_{j\neq i} A_{aj}\left(x_j^k + \epsilon \cdot x_{j\to a}^k\right) \end{aligned}$$

Using the fact that sums of the form $\sum_{j\neq i} x_j$ can be written as $\sum_j x_j - x_i$ yields:

$$\begin{aligned} x_i^{k+1} + \epsilon \cdot x_{i\to a}^{k+1} &= \eta\left(\sum_{b} A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right) - A_{ai}\left(z_a^k + \epsilon \cdot z_{a\to i}^k\right), \lambda\left(1+\hat{\tau}^k\right)\right) \\ z_a^k + \epsilon \cdot z_{a\to i}^{k+1} &= y_a - \sum_{j} A_{aj}\left(x_j^k + \epsilon \cdot x_{j\to a}^k\right) - A_{ai}\left(x_i^k + \epsilon \cdot x_{i\to a}^k\right) \end{aligned}$$

The terms $A_{ai} \cdot \epsilon \cdot z_{a\to i}^k$ and $A_{ai} \cdot \epsilon \cdot x_{i\to a}^k$ are expected to be very small since A_{ai} is small in the large system limit and $\epsilon << 1$ is small by definition and therefore these terms are also dropped. The resulting expressions then become:

$$\begin{aligned} x_i^{k+1} + \epsilon \cdot x_{i\to a}^{k+1} &\approx \eta\left(\sum_{b} A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right) - A_{ai}z_a^k, \lambda\left(1+\hat{\tau}^k\right)\right) \\ z_a^k + \epsilon \cdot z_{a\to i}^{k+1} &\approx y_a - \sum_{j} A_{aj}\left(x_j^k + \epsilon \cdot x_{j\to a}^k\right) - A_{ai}x_i^k \end{aligned}$$

Next, the expression for x^{k+1} is approximated to first order around the point $\sum_b A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right)$:

$$\begin{aligned} x_i^{k+1} + \epsilon \cdot x_{i\to a}^{k+1} &\approx \eta\left(\sum_{b} A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right), \lambda\left(1+\hat{\tau}^k\right)\right) \\ &\quad - A_{ai}z_a^k\eta'\left(\sum_{b} A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right), \lambda\left(1+\hat{\tau}^k\right)\right) \\ z_a^k + \epsilon \cdot z_{a\to i}^{k+1} &\approx y_a - \sum_{j} A_{aj}\left(x_j^k + \epsilon \cdot x_{j\to a}^k\right) - A_{ai}x_i^k \end{aligned}$$

Now by comparing the left hand side to the right hand side of the topmost expression, it appears that the only dependence on index a on the right hand side is in the second term. Thus, we can make the following identifications:

$$x_i^{k+1} = \eta\left(\sum_b A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right),\ \lambda\left(1+\hat{\tau}^k\right)\right) \tag{2.73}$$

$$\epsilon \cdot x_{i\to a}^{k+1} = -A_{ai} z_a^k \cdot \eta'\left(\sum_b A_{bi}\left(z_b^k + \epsilon \cdot z_{b\to i}^k\right),\ \lambda\left(1+\hat{\tau}^k\right)\right) \tag{2.74}$$

and similar identifications for z_a^k

$$z_a^k \approx y_a - \sum_j A_{aj}\left(x_j^k + \epsilon \cdot x_{j\to a}^k\right) \tag{2.75}$$

$$\epsilon \cdot z_{a\to i}^{k+1} \approx A_{ai} x_i^k \tag{2.76}$$

Now we can substitute eq. (2.74) into eq. (2.75) and eq. (2.76) into eq. (2.73) to get:

$$x_i^{k+1} = \eta\left(\sum_b A_{bi}\left(z_b^k + A_{bi}x_i^k\right),\ \lambda\left(1+\hat{\tau}^k\right)\right)$$

$$z_a^k = y_a - \sum_j A_{aj}\left(x_j^k - A_{aj}z_a^k \cdot \eta'\left(\sum_b A_{bj}\left(z_b^k + A_{bj}x_j^k\right),\ \lambda\left(1+\hat{\tau}^k\right)\right)\right)$$

Expanding parenthesis in the expression for x_i^{k+1} yields

$$x_i^{k+1} \approx \eta\left(\sum_b A_{bi}z_b^k + x_i^k \sum_b A_{bi}^2,\ \lambda\left(1+\hat{\tau}^k\right)\right)$$

where it is used that x_i^k is independent of b and can therefore be moved outside the sum. Furthermore, since the columns of $\boldsymbol{A}$ are assumed to have unit ℓ_2-norm, it holds that $\sum_b A_{bi}^2 = 1$. Using this and rewriting the update equation in vector form yields

$$\boldsymbol{x}^{k+1} \approx \eta\left(\boldsymbol{A}^T\boldsymbol{z}^k + \boldsymbol{x}^k,\ \lambda\left(1+\hat{\tau}^k\right)\right)$$

which is the update equation derived in [DMM10]. Next, we expand the parenthesis in the expression for z_a^k:

$$z_a^k \approx y_a - \sum_j A_{aj}x_j^k - z_a^k \sum_j A_{aj}^2 \cdot \eta'\left(\sum_b A_{bj}z_b^k + x_j^k,\ \lambda\left(1+\hat{\tau}^k\right)\right) \tag{2.77}$$

where it is used that $\sum_j A_{aj}^2 = 1$. Due to the large system limit, i.e. $m, n \to \infty$ for $\frac{m}{n} = \delta$, using the law of large numbers and the normalization of the columns of $\boldsymbol{A}$, we can make the following approximation:

$$z_a^k \sum_j A_{aj}^2 \cdot \eta'\left(\sum_b A_{bj}z_b^k + x_j^k,\ \lambda\left(1+\hat{\tau}^k\right)\right) \approx \frac{1}{m} z_a^k \sum_j \eta'\left(\sum_b A_{bj}z_b^k + x_j^k,\ \lambda\left(1+\hat{\tau}^k\right)\right)$$

Substituting this back into eq. (2.77) and rewriting the update equation in vector form yields:

$$\boldsymbol{z}^k = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^k - \frac{1}{m}\boldsymbol{z}^k \sum_j \eta' \left(\boldsymbol{A}\boldsymbol{z}^k + \boldsymbol{x}^k,\ \lambda \left(1 + \hat{\tau}^k\right)\right)$$

Writing the sum using the average operator $\langle\cdot\rangle$ and using $\frac{1}{\delta} = \frac{n}{m}$, we get the update equation in [DMM10]:

$$\boldsymbol{z}^k = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^k - \frac{1}{\delta}\boldsymbol{z}^k \left\langle \eta' \left(\boldsymbol{A}\boldsymbol{z}^k + \boldsymbol{x}^k,\ \lambda \left(1 + \hat{\tau}^k\right)\right)\right\rangle \tag{2.78}$$

Finally, from eq. (2.70) we have the update recursion for $\hat{\tau}^{k+1}$:

$$\hat{\tau}^{k+1} = \frac{\left(1 + \hat{\tau}^k\right)}{m} \sum_{i=1}^{n} \eta' \left(\sum_b A_{bi} z^k_{b\to i},\ \lambda \left(1 + \hat{\tau}^k\right)\right)$$

Using the same approximation as for x_i^k, i.e. $\sum_b A_{bi} z^k_{b\to i} \approx \sum_b A^{bi} z^k_b + x_i$, and using vector notation, we get the final update rule for $\hat{\tau}$

$$\hat{\tau}^{k+1} = \frac{\left(1 + \hat{\tau}^k\right)}{\delta} \left\langle \eta' \left(\boldsymbol{A}^T\boldsymbol{z}^k + \boldsymbol{x}^k,\ \lambda \left(1 + \hat{\tau}^k\right)\right)\right\rangle \tag{2.79}$$

Finally, defining $\gamma^k = \lambda\hat{\tau}^k$ gives rise to the following algorithm:

$$\boldsymbol{x}^{k+1} = \eta \left(\boldsymbol{A}^T\boldsymbol{z}^k + \boldsymbol{x}^k,\ \lambda \left(1 + \hat{\gamma}^k\right)\right) \tag{2.80}$$

$$\boldsymbol{z}^k = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^k - \frac{1}{\delta}\boldsymbol{z}^k \left\langle \eta' \left(\boldsymbol{A}\boldsymbol{z}^k + \boldsymbol{x}^k,\ \lambda + \gamma^k\right)\right\rangle \tag{2.81}$$

$$\gamma^k = \frac{\left(1 + \gamma^{t-1}\right)}{\delta} \left\langle \eta' \left(\boldsymbol{A}^T\boldsymbol{z}^{k-1} + \boldsymbol{x}^{k-1},\ \lambda + \gamma^{k-1}\right)\right\rangle \tag{2.82}$$

It is now seen that this algorithm only requires propagating $m + n$ messages in each iteration. Thus, the dominating operations in each iteration is the two matrix multiplications, $\boldsymbol{A}\boldsymbol{x}^k$ and $\boldsymbol{A}^T\boldsymbol{z}^k$, which both scale as $\mathcal{O}(mn)$. Therefore, each iteration of this algorithm has complexity $\mathcal{O}(mn)$. This finalizes the last step of the derivation. Note, when the update rules are considered for only one iteration, all of the approximations are expected to be negligible in the large system limit. But there is no theoretical guarantee that the errors do accumulate over multiple iterations. However, Donoho et al. claim that it is highly unlikely and it has not been observed despite massive numerical simulation studies.

Algorithm 1 AMP algorithm (AMP)

```
• Initialization: Set $t = 0, \boldsymbol{x} = 0$, $\tau^k = 1$ and $\boldsymbol{z} = \boldsymbol{y}$.
• repeat until stopping criteria:
        $\boldsymbol{x}^k = \eta\left(\boldsymbol{A}^T\boldsymbol{z}^{k-1} + \boldsymbol{x}^{k-1},\ \lambda + \gamma^{k-1}\right)$
        $\boldsymbol{z}^k = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^k - \frac{1}{\delta}\boldsymbol{z}^{k-1}\left\langle \eta'\left(\boldsymbol{A}\boldsymbol{z}^{k-1} + \boldsymbol{x}^{k-1},\ \lambda + \gamma^{k-1}\right)\right\rangle$
        $\hat{\gamma}^k = \frac{(1+\hat{\gamma}^{k-1})}{\delta}\left\langle \eta'\left(\boldsymbol{A}^T\boldsymbol{z}^k + \boldsymbol{x}^k,\ \lambda + \gamma^{k-1}\right)\right\rangle$
        Increase $k$
```

The algorithm is summarized in Algorithm 1. Following the notation in [DMM10], the algorithm will be referred to as AMP0 for $\lambda = 0$ and AMPA for $\lambda > 0$.

Comparing the algorithm to the standard form of iterative thresholding methods (see literature review in section 1.2), it is seen that AMP distinguishes itself by the last term in the update equation for the residuals, i.e. the term:

$$\frac{1}{\delta}\boldsymbol{z}^{k-1}\left\langle \eta'\left(\boldsymbol{A}\boldsymbol{z}^{k-1} + \boldsymbol{x}^{k-1},\ \lambda + \gamma^{k-1}\right)\right\rangle \tag{2.83}$$

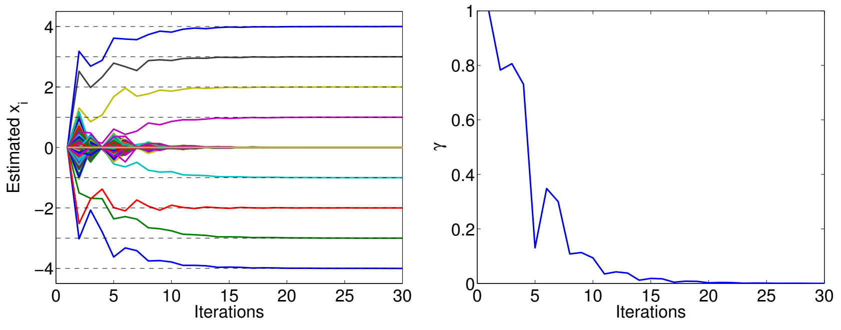
Donoho et al. states that this term leads to a substantial increase in recovery without increasing the computational complexity significantly [DMM10]. This term can be interpreted as a momentum term or the Onsager reaction term from statistical physics [DMM09].

Example: Toy Problem

The AMP0 algorithm is now illustrated using a small toy example. Consider a noiseless problem with $n = 1000$, $m = 200$, and $k = 8$. That is, a linear inverse problem with 1000 unknowns, 200 equations and the true solutions has 8 non-zero elements. Let the true solution $\boldsymbol{x}_0$ be:

$$\boldsymbol{x}_0 = \begin{bmatrix} -4 & -3 & -2 & -1 & 1 & 2 & 3 & 4 & 0 & 0.. \end{bmatrix}^T \in \mathbb{R}^n \tag{2.84}$$

The measurements $\boldsymbol{y}$ are then generated using $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$, where $A_{ij} \sim \mathcal{N}(0, 1/m)$. Figure 2.7 shows the result of applying AMP0 to this problem. In particular, figure (a) shows the estimated coefficients $\hat{\boldsymbol{x}}$ as a function of the iteration number. The dashed lines indicates the true coefficients. The estimated coefficients are initialized at 0 in the first iteration and then they converge to their respective true values in approximately 15 iterations. Figure (b) shows the evolution of the threshold parameter γ as a function of the iteration number.



(a) Estimated coefficients (b) γ

Figure 2.7: Illustration of the AMP0 algorithm using a noiseless toy problem with $n = 1000$, $\delta = 0.2$, $k = 8$. (a) The estimated coefficients as a function of iterations. The dashed lines indicate the true values. (b) The threshold parameter γ_k as a function of iterations.

The State Evolution Fxormalism

Although not considered in this thesis, another very interesting aspect of the AMP algorithm is the associated state evolution (SE) formalism [BM10, DMM10, DMM09]. That is, the parameter τ_k^2 (as in eq. (2.79)) can be considered as the state of the algorithm and it turns out that this state behaves in a predictable manner in the large system limit. For instance, consider using the AMP0 algorithm on a noiseless problem of the form $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. Then the state variable τ_k^2 is accurately predicted by the following analytical expressions:

$$\tau_{k+1}^2 = \frac{1}{\delta}\mathbb{E}\left[\eta\left(X_0 + \tau_k Z,\, \gamma_k\right) - X_0\right]^2 \tag{2.85}$$

$$\gamma_{k+1} = \frac{\gamma_k}{\delta}\mathbb{E}\left[\eta'\left(X_0 + \tau_k Z,\, \gamma_k\right)\right] \tag{2.86}$$

where X_0 is a random variable distributed according to the true prior distribution of $\boldsymbol{x}$ and $Z \sim \mathcal{N}(0, 1)$. For a Gaussian $\boldsymbol{A}$ matrix, the fixed points of the recursion in eq. (2.85) can be used to predict whether the algorithm can solve the current problem or not [DMM10]. In fact, Donoho et al. derives the phase transition curve (see literature review in section 1.2) for the AMP algorithm based on this state evolution formalism. This phase transition curve, $\rho_{SE}(\delta)$, is shown to be identical to that derived from combinatorial geometry (CG) for ℓ_1-minimization:

$$\rho_{SE}(\delta) = \rho_{CG}(\delta) = \max_{z \geq 0} \frac{1 - (2/\delta)\left[(1 + z^2)\Phi(-z) - z\phi(z)\right]}{1 + z^2 - 2\left[(1 + z^2)\Phi(-z) - z\phi(z)\right]} \tag{2.87}$$

where $\phi(z)$ and $\Phi(z)$ are the density and cumulative distribution of a standardized Gaussian variable z, respectively. Moreover, they show that the theoretical quantities agrees with empirical simulation.

2.3 Generalized Approximate Message Passing

The AMP algorithm introduced in last section provide a method for solving the BP or BPDN problem in an efficient manner. Sundeep Rangan has shown that this framework can be generalized to handle essentially arbitrary prior distributions and arbitrary noise distributions. The only requirement for the two sets of distributions are that they factorize. This generalized framework, *Generalized Approximate Message Passing (GAMP)*, is introduced in the paper [Ran10]. The flexibility of GAMP allows us to do efficient inference using sparsity promoting prior distributions like the spike and slab prior [IR05]. Furthermore, since the noise distribution is also arbitrary, the framework can also be used for classification by using a binomial noise distribution [ZS14], but this is not considered in this work, though.

Even though the flexibility of the model is greatly increased, the computational complexity remains the same, namely $\mathcal{O}(nm)$. The GAMP framework can both be configured to perform MAP estimation based on max-sum message passing and it can be configured to perform MMSE estimation based on sum-product message passing. The derivation of the GAMP framework is somewhat more straightforward than the derivation of AMP. Here the derivation is mainly based on Taylor approximations and an application of the Central Limit Theorem.

The GAMP algorithm is stated in Algorithm 2 in its most general form. However, Rangan also provides a simplified version of this algorithm, where the individual variance parameters τ_i^r are exchanged for a common variance parameter τ^r and similar for τ_i^x, τ_a^s and τ_a^p. This simplified version is listed in Algorithm 3. The first algorithm is referred to as GAMP1 algorithm, whereas the latter is referred to as the GAMP2 algorithm. It can be shown the AMP algorithm introduced by Donoho et al. is actually a special case of the GAMP algorithm. In fact, the GAMP2 algorithm correspond to the AMP algorithm if the prior and noise distribution is chosen to be Laplace and Gaussian, respectively (see Appendix B.2 for more details).

Consider now the computational complexity of the two algorithms. Assuming the scalar functions and their derivatives have closed form expressions, the GAMP1 algorithm is dominated by two matrix multiplication involving $\boldsymbol{A}$ and two matrix multiplications involving $\boldsymbol{A}^T$. Due to the scalar variances, the GAMP2 algorithm only requires one multiplication of $\boldsymbol{A}$ and one multiplication of $\boldsymbol{A}^T$. Thus, both GAMP1 and GAMP2 scale as $\mathcal{O}(mn)$, but the proportionality constant of GAMP2 is half of the proportionality constant for GAMP1, which can have a large impact for large systems of equations.

Before we dive into the derivation of the GAMP framework, we will spent a

Algorithm 2 GAMP algorithm (GAMP1)

```
• Initialization:
        Set $k = 0$ and $\hat{s}_a(-1) = 0$.
        Set $\hat{x}_j^0$ and $\tau_j^0(k)$ based on type of inference (MAP/MMSE).
• repeat:
        Step 1. For each $a \in [m]$:    $z_a^k = \sum_j A_{aj}\hat{x}_j^k$
                                         $\tau_a^p(k) = \sum_j A_{aj}^2 \tau_j^x(k)$
                                         $\hat{p}_a^k = z_a^k - \tau_a^p(k)\hat{s}_a^{k-1}$
        Step 2. For each $a \in [m]$:    $\hat{s}_a^k = g_{\text{out}}\left(\hat{p}_a^k,\, y_a,\, \tau_a^p(k)\right)$
                                         $\tau_a^s(k) = -\frac{\partial}{\partial \hat{p}} g_{\text{out}}\left(\hat{p}_a^k,\, y_a,\, \tau_a^p(k)\right)$
        Step 3: For each $i \in [n]$:    $\tau_i^r(k) = \left(\sum_a A_{ai}^2 \tau_a^s(k)\right)^{-1}$
                                         $\hat{r}_i^k = \hat{x}_i^k + \tau_i^r(k)\sum_a A_{ai}\hat{s}_a^k$
        Step 4: For each $i \in [n]$:    $\hat{x}_i^{k+1} = g_{\text{in}}\left(\hat{r}_j^k, q_j, \tau_i^r(k)\right)$
                                         $\tau_i^x(k+1) = \tau_j^r(k)\frac{\partial}{\partial \hat{r}} g_{\text{in}}\left(\hat{r}_j^k, q_j, \tau_i^r(k)\right)$
```

few moments discussing the algorithm itself and the involved quantities. Suppose that the individual elements in $\boldsymbol{x}$ are independently distributed according to $p(x_i|q_i)$, where q_i is a known hyperparameter. Similarly, suppose that the measurements are independently distributed according to $p(y_a|\boldsymbol{x})$. The core of the algorithm is what Rangan calls the two *scalar functions*: $g_{\text{in}}(\cdot)$ and $g_{\text{out}}(\cdot)$. As we will see soon, these two functions depend on the functional forms of the prior distribution $p(x_i|q_i)$ and noise distribution $p(y_a|\boldsymbol{x})$ and whether we want to do MAP inference or MMSE inference. Thus, these two functions control the basic behaviour of the algorithm. As stated earlier, the GAMP framework can handle essentially any prior and noise distribution. However, for the algorithms to remain efficient, it is necessary that the scalar functions can be expressed in closed form, which limits the range of applicable distributions a bit.

In the GAMP algorithm listed in Algorithm 2, $\hat{x}_i^k$ denotes that estimate of the i'th element of $\boldsymbol{x}$ in the k'th iteration and $\tau_i^x(k)$ can be interpreted as the associated uncertainty of $\hat{x}_i^k$. In fact, when the algorithm is running in MMSE mode, $\tau_i^x(k)$ can be interpreted as an approximation of the posterior variance of

Algorithm 3 GAMP algorithm w. scalar variances (GAMP2)

```
1. Initialization:
        Set $k = 0$ and $\hat{s}_a(-1) = 0$.
        Set $\hat{x}^0$ and $\tau^0(k)$ based on type of inference (MAP/MMSE).
2. repeat:
        Step 1. For each $a \in [m]$:    $\tau^p(k) = \left(\frac{1}{m}\right) ||\boldsymbol{A}||_F^2 \, \tau^x(k)$
                                         $\hat{p}_a^k = \sum_j A_{aj} \hat{x}_j^k - \tau^p(k) \hat{s}_a^{k-1}$
        Step 2. For each $a \in [m]$:    $\hat{s}_a^k = g_{\text{out}}\left(\hat{p}_a^k,\, y_a,\, \tau^p(k)\right)$
                                         $\tau^s(k) = -\frac{1}{m} \sum_{a=1}^{m} \frac{\partial}{\partial \hat{p}} g_{\text{out}}\left(\hat{p}_a^k,\, y_a,\, \tau^p(k)\right)$
        Step 3: For each $i \in [n]$:    $\frac{1}{\tau^r(k)} = \left(\frac{1}{n}\right) ||\boldsymbol{A}||_F^2 \, \tau^s(k)$
                                         $\hat{r}_i^k = \hat{x}_i^k + \tau^r(k) \sum_a A_{ai} \hat{s}_a^k$
        Step 4: For each $i \in [n]$:    $\hat{x}_i^{k+1} = g_{\text{in}}\left(\hat{r}_j^k, q_j, \tau^r(k)\right)$
                                         $\tau^x(k+1) = \tau^r(k) \frac{\partial}{\partial \hat{r}} g_{\text{in}}\left(\hat{r}_j^k, q_j, \tau^r(k)\right)$
```

$\hat{x}_i^k$. The scalar functions for both MAP and MMSE estimation are summarized in table 2.2.

Interpretation of the Scalar Functions for MAP Estimation

To use the GAMP algorithm for MAP inference, the input scalar function g_{in} is given as:

$$g_{\text{in}}\left(\hat{r}, q, \tau^r\right) = \arg\max_x F_{\text{in}}\left(x, \hat{r}, q, \tau^r\right), \tag{2.88}$$

where

$$F_{\text{in}}\left(x, \hat{r}, q, \tau^r\right) \equiv f_{\text{in}}\left(x, q\right) - \frac{1}{2\tau^r}\left(x - \hat{r}\right)^2 \tag{2.89}$$

The function $f_{\text{in}}(x, q)$ is the logarithm of the prior density, i.e. $f_{\text{in}}\left(x, q\right) = \log\left[p(x|q)\right]$. Therefore, $g_{\text{in}}\left(\hat{r}, q, \tau^r\right)$ can interpreted as the MAP estimate under

Table 2.2: Scalar functions for both MAP and MMSE estimation.

Scalar function	**MAP**	**MMSE**
$\hat{z}_0$	$\arg\max_z F_{\text{out}}\left(z, \hat{p}, y, \tau^p\right)$	$\mathbb{E}\left[z \middle\| \hat{p}, y, \tau^p\right]$
$g_{\text{out}}\left(\hat{p}, y, \tau^p\right)$	$\frac{1}{\tau^p}\left(\hat{z}_0 - \hat{p}\right)$	$\frac{1}{\tau^p}\left(\hat{z}_0 - \hat{p}\right)$
$-g'_{\text{out}}\left(\hat{p}, y, \tau^p\right)$	$\frac{f''_{\text{out}}(\hat{z}_0, y)}{\tau^p f''_{\text{out}}(\hat{z}_0, y) - 1}$	$\left(\frac{\tau^p - \mathbb{V}\left[z \middle\| \hat{p}, y\right]}{\tau^p}\right)^2$
$g_{\text{in}}\left(\hat{r}, q, \tau^r\right)$	$\arg\max_x F_{\text{in}}\left(z, \hat{r}, q, \tau^r\right)$	$\mathbb{E}\left[x \middle\| \hat{r}, q, \tau^r\right]$
$\tau^r g'_{\text{in}}\left(\hat{r}, q, \tau^r\right)$	$\frac{\tau^r}{1 - \tau^r f''_{\text{in}}(\hat{x}, q)}$	$\mathbb{V}\left[x \middle\| \hat{r}, q, \tau^r\right]$

the (unnormalized) posterior distribution given by:

$$p(x) \propto \exp\left[F_{\text{in}}\left(x, \hat{r}, q, \tau^r\right)\right] \tag{2.90}$$

In order words, we can interpret $\hat{r}$ as a noise corrupted version of x. For MAP inference, the output function $g_{\text{out}}\left(\hat{p}, y, \tau^p\right)$ is given by

$$g_{\text{out}}\left(\hat{p}, y, \tau^p\right) = \frac{1}{\tau^p}\left(\hat{z}_0 - \hat{p}\right), \qquad \hat{z}_0 = \arg\max_z F_{\text{out}}\left(z, \hat{p}, y, \tau^p\right) \tag{2.91}$$

where

$$F_{\text{out}}\left(z, \hat{p}, y, \tau^p\right) \equiv f_{\text{out}}\left(z, y\right) - \frac{1}{2\tau^p}\left(z - \hat{p}\right)^2, \tag{2.92}$$

where the function $f_{\text{out}}(y, z)$ is the logarithm of the noise distribution $p(y_a|z_a)$ and z_a is the noise free output, i.e. $z_a = \left(\boldsymbol{Ax}\right)_a$. Thus, $\hat{z}^0$ can be interpreted as the MAP estimate of a random variable Z given $Y = y$, where $Z \sim \mathcal{N}\left(\hat{p}, \tau^p\right)$ and $Y \sim p(y|z)$. For MAP estimation, the variables $\hat{x}_i$ and τ_i^x should be initialized according to:

$$\hat{x}_i^0 = \arg\max_{x_i} f_{\text{in}}(x_i, q_i) \qquad\qquad \tau_i^x(0) = \frac{1}{f''_{\text{in}}(\hat{x}_i^0, q_i)} \tag{2.93}$$

Interpretation of Scalars Function for MMSE Estimation

The GAMP algorithm can also provide approximative posterior distributions for $p(x_i|\boldsymbol{y})$ and $p(z_a|\boldsymbol{y})$, which in turn can be used for MMSE inference, i.e.

$$x_i^{MMSE} = \int x_i \cdot p(x_i|q, \boldsymbol{y})\,\mathrm{d}x_i \tag{2.94}$$

and similar for z_a. The GAMP approximation for the posterior distribution of x_i is given by:

$$p\left(x_i|q,\boldsymbol{y}\right) = \frac{p(x_i|q)\mathcal{N}(x_i|\hat{r},\tau^r)}{\int p(x_i|q)\mathcal{N}(x_i|\hat{r},\tau^r)\,\mathrm{d}x_i} \tag{2.95}$$

As we will see soon, the input scalar function g_{in} for MMSE estimation is simply the conditional expectation of x_i under this distribution, i.e.:

$$g_{\text{in}}\left(\hat{r},q,\tau^r\right) = \mathbb{E}\left[x\middle|\hat{r},q,\tau^r\right] = x_i^{MMSE} \tag{2.96}$$

The scaled partial derivative of $\tau^r g'_{\text{in}}\left(\hat{r},q,\tau^r\right)$ w.r.t. $\hat{r}$ is then the conditional variance under this distribution:

$$\tau^r g'_{\text{in}}\left(\hat{r},q,\tau^r\right) = \mathbb{V}\left[\boldsymbol{x}\middle|\hat{r},q,\tau^r\right] \tag{2.97}$$

Analogously, the posterior distribution of z_a is approximated by:

$$p\left(z_a\middle|y_a,q\right) = \frac{p(y_a|z_a,q)\mathcal{N}\left(z_a\middle|\hat{p},\tau^p\right)}{\int p(y_a|z_a,q)\mathcal{N}\left(z_a\middle|\hat{p},\tau^p\right)\,\mathrm{d}z_a} \tag{2.98}$$

The output scalar function is related to the conditional expectation of z_a under this distribution:

$$g_{\text{out}}\left(\hat{p},y,\tau^p\right) = \frac{1}{\tau^p}\left(\hat{z}_0-\hat{p}\right), \qquad \hat{z}_0 = \mathbb{E}\left[z_a\middle|\hat{p},y,\tau^p\right]. \tag{2.99}$$

and the partial derivative of $g_{\text{out}}\left(\hat{p},y,\tau^p\right)$ is related to the conditional variance in a similar way. For MMSE estimation, the variables $\hat{x}_i$ and τ_i^x should be initialized according to:

$$\hat{x}_i^0 = \mathbb{E}\left[x_i|q_i\right] \qquad\qquad \tau_i^x(0) = \mathbb{V}\left[x_i|q_i\right] \tag{2.100}$$

That is, $\hat{x}_i^0$ and $\tau_i^x(0)$ are initialized as the mean and variance of the prior distribution.

The next two sections describe derivation of the GAMP framework using both the max-sum algorithm and the sum-product algorithm. Since the two derivations have a large number of similarities, the derivation using max-sum is carried out in detail, while for the sum-product case, only the differences are described.

Derivation of Max-Sum Algorithm for MAP Estimation

The purpose of this section is to derive the update equations for max-sum algorithm and argue that the scalar functions $g_{\text{in}}(\cdot)$ and $g_{\text{out}}(\cdot)$ are determined by the prior distribution and the noise distribution, respectively. The derivation given here follows the approach in [Ran10]. In the remainder of this chapter, it is assumed that the columns of $\boldsymbol{A}$ are scaled such to have variance $\frac{1}{m}$.

Both the prior distribution and the noise distribution have to factorize. That is, the prior distribution on $\boldsymbol{x}$ have to have the form

$$p(\boldsymbol{x}|\boldsymbol{q}) = \prod_{i=1}^{n} p(x_i|q_i), \tag{2.101}$$

where $\boldsymbol{q}$ are hyperparameters. The same holds true for the noise distribution, which is given by

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{a=1}^{m} p(y_a|\boldsymbol{x}). \tag{2.102}$$

The joint probability distribution can then be written as:

$$\begin{aligned} p\left(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{q}\right) &= p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{q}) \\ &= \prod_{a=1}^{m} p(y_a|\boldsymbol{x}) \prod_{i=1}^{n} p\left(x_i|q_i\right) \end{aligned} \tag{2.103}$$

Using this decomposition, the corresponding factor graph can be set up. Due to the use of the max-sum algorithm, the factors in the factor graph correspond to the logarithm of the factor functions in the decomposition in eq. (2.103). The logarithm of the i'th prior distribution is denoted $f_{\text{in},i}$ and the logarithm of the a'th noise distribution is $f_{\text{out},a}$. That is,

$$f_{\text{in},i}(x_i) \equiv \ln p(x_i|q_i) \tag{2.104}$$

$$f_{\text{out},a}(y_a, z_a) \equiv \ln p(y_a|z_a), \tag{2.105}$$

where the auxiliary variable z_a is defined as $z_a = \left(\boldsymbol{A}\boldsymbol{x}\right)_a$, i.e. $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$. The resulting factor graph is depicted in figure 2.8. As before, the factor graph contains loops and therefore it is necessary to resort to loopy message passing. In the next section, the max-sum update rules are derived based on this factor graph.

Exact Max-Sum Message Passing Equations

We start from the leaf nodes and propagate messages towards to center of the graph. The messages from the right-most leaf node, i.e. factor node $f_{\text{out},a}$, to

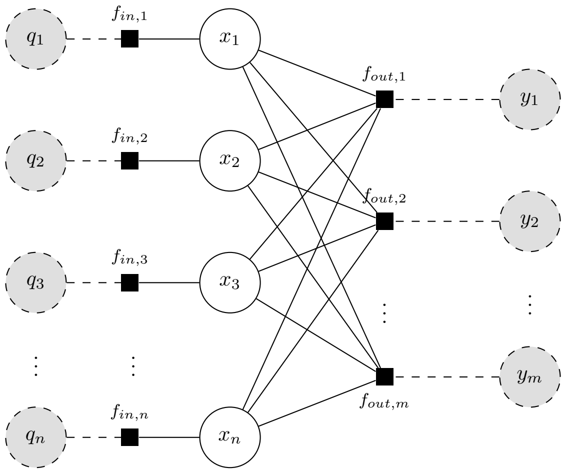


Figure 2.8: Factor graph for GAMP model defined in eq. (2.103)

variable node x_i is given by:

$$\mu_{f_{\text{out},a}\to x_i}(x_i) = \max_{\boldsymbol{x}\setminus x_i}\left[f_{\text{out}}(y_a,z_a) + \sum_{j\neq i}\mu_{x_j\to f_{\text{out},a}}(x_j)\right] \tag{2.106}$$

where the maximization is over the set of variables $\boldsymbol{x}\setminus x_i \equiv \{x_j : j\in[n], j\neq i\}$. Next, the messsage from the left-most leaf is considered. That is, the message from factor node $f_{\text{in},i}$ to variable node x_i:

$$\mu_{f_{\text{in},i}\to x_i}(x_i) = f_{\text{in}}(x_i, q_i), \tag{2.107}$$

Finally, the message from variable node x_i to factor node $f_{\text{out},a}$ is given by:

$$\begin{aligned}\mu_{x_i\to f_{\text{out},a}}(x_i) &= \mu_{f_{\text{in},i}\to x_i}(x_i) + \sum_{b\neq a}\mu_{f_{\text{out},b}\to x_i}(x_i)\\ &= f_{\text{in}}(x_i,q_i) + \sum_{b\neq a}\mu_{f_{\text{out},b}\to x_i}(x_i)\end{aligned} \tag{2.108}$$

Again, the two messages in eq. (2.106) and eq. (2.108) reveal the problem with message passing in graphs with loops. Resorting to loopy message gives rise to the following update equations:

$$\mu^k_{a\to i}(x_i) = \max_{\boldsymbol{x}\setminus x_i}\left\{f_{\text{out}}(z_a,y_a) + \sum_{j\neq i}\mu^k_{j\to a}(x_j)\right\} \tag{2.109}$$

and

$$\mu_{i\to a}^{k+1}(x_i) = f_{\text{in}}(x_i, q_i) + \sum_{b\neq a} \mu_{b\to i}^{k}(x_i) \tag{2.110}$$

where k is the iteration index and $z_a = (\boldsymbol{Ax})_a$. Note, that both types of messages are (unnormalized) functions on the entire real line. In general, additive constants are not important, since we are dealing with logarithmic messages. In the next section a series of approximation are introduced to simplified this message passing scheme.

Approximation of the Max-Sum Messages

As stated above, the messages in eq. (2.109) and (2.110) are functions defined on the entire real line. We will now introduce a set of approximations, which reduces these messages to a few parameters.

First, define $\hat{x}_{i\to a}^{k}$ as the value that maximizes the message from variable node x_i to factor node $f_{\text{out},a}$ in the k'th iteration, i.e.

$$\hat{x}_{i\to a}^{k} \equiv \arg\max_{x_i} \mu_{i\to a}^{k}(x_i) \tag{2.111}$$

The terms $\mu_{j\to a}^{k}(x_j)$ in eq. (2.109) are now approximated using a second order Taylor approximation around the point $\hat{x}_{j\to a}^{k}$, i.e. around its maximum. This is reasonable because the values of x_j in the maximization in eq. (2.109) will be close to $\hat{x}_{j\to a}^{k}$ for small values of A_{ai} due to $z_a = \sum_i A_{ai}x_i$. This approximation yields:

$$\begin{aligned}
\mu_{j\to a}^{k}(x_j) &\approx \mu_{j\to a}^{k}(\hat{x}_{j\to a}^{k}) + \frac{\partial}{\partial x_j}\mu_{j\to a}^{k}(x_j)\,\big|_{x_j=\hat{x}_{j\to a}^{k}}(x_j - \hat{x}_{j\to a}^{k}) \\
&\quad + \frac{1}{2}\frac{\partial^2}{\partial x_j^2}\mu_{j\to a}^{k}(x_j)\,\big|_{x_j=\hat{x}_{j\to a}^{k}}(x_j - \hat{x}_{j\to a}^{k})^2 \\
&= \mu_{j\to a}^{k}(\hat{x}_{j\to a}^{k}) + \frac{1}{2}\frac{\partial^2}{\partial x_j^2}\mu_{j\to a}^{k}(x_j)\,\big|_{x_j=\hat{x}_{j\to a}^{k}}(x_j - \hat{x}_{j\to a}^{k})^2 \\
&= \mu_{j\to a}^{k}(\hat{x}_{j\to a}^{k}) - \frac{1}{2\tau_{j\to a}^{x}}\left(x_j - \hat{x}_{j\to a}^{k}\right)^2,
\end{aligned} \tag{2.112}$$

where it is used that the first partial derivative is zero, when evaluated at the maxima. The following definition is used in the above approximation:

$$\frac{1}{\tau_{j\to a}^{x}} \equiv -\frac{\partial^2}{\partial x_j^2}\mu_{j\to a}^{k}(x_j)\,\big|_{x_j=\hat{x}_{j\to a}^{k}} \tag{2.113}$$

Thus, $\tau^x_{j\to a}$ plays the role of the reciprocal negative curvature of the message from variable node x_j to factor node $f_{\text{out},a}$ evaluated at its maxima. Note, that the quantity $\tau^x_{j\to a}$ does also depend on the iteration number k, but to keep the notation uncluttered, it is omitted if is it not strictly necessary.

An additional approximation in now introduced by assuming that $\tau^x_{j\to a}$ is independent of a. That is, $\tau^x_{j\to a} = \tau^x_j$ for all a. Using this assumption, the messages become:

$$\mu^k_{j\to a}(x_j) \approx \mu^k_{j\to a}(\hat{x}^k_{j\to a}) - \frac{1}{2\tau^x_j}\left(x_j - \hat{x}^k_{j\to a}\right)^2 \tag{2.114}$$

The expression for the messages in eq. (2.114) is now substituted into eq. (2.109):

$$\mu^k_{a\to i}(x_i) \approx \max_{\boldsymbol{x}\setminus x_i}\left\{f_{\text{out}}(z_a, y_a) + \sum_{j\neq i}\left[\mu^k_{x_j\to g_a}(\hat{x}^k_{j\to a}) - \frac{1}{2\tau^x_j}\left(x_j - \hat{x}_{j\to a}\right)^2\right]\right\}$$

The terms $\mu^k_{j\to a}(\hat{x}^k_{j\to a})$ do not depend on x_i and can be absorbed by the normalization constant. Therefore, the message simplifies to:

$$\mu^k_{a\to i}(x_i) \approx \max_{\boldsymbol{x}\setminus x_i}\left\{f_{\text{out}}(z_a, y_a) - \sum_{j\neq i}\frac{1}{2\tau^x_j}\left(x_j - \hat{x}^k_{j\to a}\right)^2\right\} \tag{2.115}$$

This optimization problem is now further simplified using a two step procedure. The first step is to optimize to sum-term w.r.t. x_j for $j \neq i$ subject to the constraint $z_a = A_{ai}x_i + \sum_{j\neq i} A_{aj}x_j$, but for fixed values of x_i and z_a. The second step is then to optimize the result w.r.t. z_a.

To solve the first step, assume x_i and z_a are fixed. Then the corresponding optimization problem is given by:

$$J = \min_{\boldsymbol{x}}\sum_{j\neq i}\frac{1}{2\tau^x_j}\left(x_j - \hat{x}^k_{j\to a}\right)^2 \quad \text{subject to} \quad z_a = A_{ai}x_i + \sum_{j\neq i} A_{aj}x_j, \tag{2.116}$$

which is recognized as a least squares problem with an equality constraint. Such a problem can be solved by introducing a Lagrange multiplier and forming the Lagrangian function [NW06]:

$$f(\boldsymbol{x}, \lambda) = \sum_{j\neq i}\frac{1}{2\tau^x_j}\left(x_j - \hat{x}^k_{j\to a}\right)^2 + \lambda\left(z_a - A_{ai}x_i - \sum_{j\neq i} A_{aj}x_j\right) \tag{2.117}$$

The procedure is now to compute the partial derivatives of $f(\boldsymbol{x}, \lambda)$ in eq. (2.117), equating them to zero and solve the resulting system of equations. The long and tedious computation is shown in appendix B.1, but here we skip straight to the result[4]:

$$J = \frac{1}{2} \frac{1}{\sum_{j \neq i} A_{aj}^2 \tau_j^x} \left(z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} \hat{x}_{j \to a}^k \right)^2$$

By introducing the following quantities:

$$\hat{p}_{a \to i} \equiv \sum_{j \neq i} A_{aj} \hat{x}_{j \to a}^k \qquad \hat{\tau}_{a \to i}^p \equiv \sum_{j \neq i} A_{aj}^2 \tau_j^x, \tag{2.118}$$

the solution of the least squares optimization problem can be written as:

$$J = \frac{1}{2 \hat{\tau}_{a \to i}^p} \left(z_a - A_{ai} x_i - \hat{p}_{a \to i} \right)^2 \tag{2.119}$$

We have now solved the first of the two optimization steps. To solve the second step, we insert the above result into eq. (2.115) and then maximize over z_a

$$\mu_{a \to i}^k (x_i) \approx \max_{z_a} \left\{ f_{\text{out}} (z_a, y_a) - \frac{1}{2 \hat{\tau}_{a \to i}^p} \left(z_a - \hat{p}_{a \to i} - A_{ai} x_i \right)^2 \right\} \tag{2.120}$$

Now, by defining the function:

$$H (\hat{p}, y, \tau^p) = \max_{z} \left\{ f_{\text{out}} (z, y) - \frac{1}{2 \tau^p} (z - \hat{p})^2 \right\}, \tag{2.121}$$

the message from factor node $f_{\text{out},a}$ to variable node x_i can be written as:

$$\mu_{a \to i}^k (x_i) \approx H \left(\hat{p}_{a \to i} + A_{ai} x_i ,\, y_a ,\, \hat{\tau}_{a \to i}^p \right) \tag{2.122}$$

We will now strive to simplify these messages even further. In particular, the goal is to simplify the first argument in the function $H(\cdot)$ above. In order to do that, we first introduce a few new definitions:

$$\hat{p}_a \equiv \sum_i A_{ai} \hat{x}_{i \to a} \qquad \tau_a^p \equiv \sum_i A_{ai}^2 \tau_i^x \tag{2.123}$$

Notice that these two new quantities do not depend on index i. Using these new definitions, we can rewrite the expression for $\hat{p}_{a \to i}$ as:

$$\hat{p}_{a \to i} = \sum_{r \neq i} A_{ar} \hat{x}_{r \to a} = \sum_r A_{ar} \hat{x}_{r \to a} - A_{ai} \hat{x}_{i \to a} = \hat{p}_a - A_{ai} \hat{x}_{i \to a} \tag{2.124}$$

[4]There is a typo in the solution to this least squares problem in the paper [Ran10]. The expression for J just below eq. (106) contains a summation operator, which is should not.

and $\hat{\tau}^p_{a\to i}$ as:

$$\hat{\tau}^p_{a\to i} = \sum_{r\neq i} A^2_{ar}\tau^x_r = \sum_j A^2_{aj}\tau^x_j - A^2_{ai}\tau_i = \tau^p_a - A^2_{ai}\tau^x_i \tag{2.125}$$

The results from eq. (2.124) and eq. (2.125) are now substituted into eq. (2.122):

$$\mu^k_{a\to i}(x_i) \approx H\left(\hat{p}_a - A_{ai}\hat{x}^k_{i\to a} + A_{ai}x_i,\, y_a,\, \tau^p_a - A^2_{ai}\tau^x_i\right) \tag{2.126}$$

Now two new approximations are introduced. First, since the columns of $\boldsymbol{A}$ are assumed to have variance $\frac{1}{m}$, the elements A^2_{ai} are expected to be small and therefore we neglect the term $A^2_{ai}\tau^x_i$. Moreover, we will make the following approximation: $\hat{x}^k_{i\to a} = \hat{x}^k_i$. Applying these two approximations yields:

$$\begin{aligned}\mu^k_{a\to i}(x_i) &\approx H\left(\hat{p}_a - A_{ai}\hat{x}^k_i + A_{ai}x_i,\, y_a,\, \tau^p_a\right)\\ &= H\left(\hat{p}_a + A_{ai}\left(x_i - \hat{x}^k_i\right),\, y_a,\, \tau^p_a\right)\end{aligned} \tag{2.127}$$

We will now introduce yet another approximation. That is, the expression in eq. (2.127) is approximated by a second order expansion[5] of eq. (2.127) around the point $\hat{p}_a$:

$$\begin{aligned}\mu^k_{a\to i}(x_i) \approx{}& H\left(\hat{p}, y_a, \tau^p_a\right)\big|_{\hat{p}=\hat{p}_a} + \frac{\partial H\left(\hat{p}, y_a, \tau^p_a\right)}{\partial \hat{p}}\Big|_{\hat{p}=\hat{p}_a}\left[A_{ai}\left(x_i - \hat{x}^k_i\right)\right]\\ &+ \frac{1}{2}\frac{\partial^2 H\left(\hat{p}, y_a, \tau^p_a\right)}{\partial \hat{p}^2}\Big|_{\hat{p}=\hat{p}_a}\left[A_{ai}\left(x_i - \hat{x}^k_i\right)\right]^2\end{aligned} \tag{2.128}$$

The first term $H\left(\hat{p}, y_a, \tau^p_a\right)\big|_{\hat{p}=\hat{p}_a}$ is constant w.r.t. x_i and can therefore be absorbed into the constant:

$$\begin{aligned}\mu^k_{a\to i}(x_i) \approx{}& \frac{\partial H\left(\hat{p}, y_a, \tau^p_a\right)}{\partial \hat{p}}\Big|_{\hat{p}=\hat{p}_a}\left[A_{ai}\left(x_i - \hat{x}^k_i\right)\right]\\ &+ \frac{1}{2}\frac{\partial^2 H\left(\hat{p}, y_a, \tau^p_a\right)}{\partial \hat{p}^2}\Big|_{\hat{p}=\hat{p}_a}\left[A_{ai}\left(x_i - \hat{x}^k_i\right)\right]^2\end{aligned} \tag{2.129}$$

It turns out that the first and second partial derivatives are closely related to one of the scalar functions in the algorithm, namely $g_{\text{out}}(\cdot)$. But in order to see this, we first need small detour to figure out how to actually compute these partial derivatives.

To compute these partial derivatives, Sundeep Rangan uses the following results[6]. Let $f : \mathbb{R} \to \mathbb{R}$ be a function, let $r, \tau \in \mathbb{R}$ be scalars and let $k \in \mathbb{N}$ be

[5] In the paper [Ran10], this approximation is described as a first order approximation.

[6] Rangan points out that Λf in eq. (2.132) can be interpreted as a quadratic variant of the Legendre transformation [ZRM09] of f [Ran10].

natural number, then define the following functions:

$$(Lf)(x,r,\tau) \equiv f(x) - \frac{1}{2\tau}(r-x)^2 \tag{2.130}$$

$$(\Gamma f)(r,\tau) \equiv \arg\max_x (Lf)(x,r,\tau) \tag{2.131}$$

$$(\Lambda f)(r,\tau) \equiv \max_x (Lf)(x,r,\tau) \tag{2.132}$$

$$\left(\Lambda^{(k)} f\right)(r,\tau) \equiv \frac{\partial^k}{\partial r^k}(\Lambda f)(r,\tau), \tag{2.133}$$

Now assume that f is twice differentiable and the above maximizations exists and are unique. Then by defining $\hat{x} = (\Gamma f)(r,\tau)$ and by using the above definitions, it can be shown (straightforward proofs are given in the paper [Ran10]) that the following holds:

$$\hat{x} = (\Gamma f)(r,\tau) \tag{2.134}$$

$$\left(\Lambda^{(1)} f\right)(r,\tau) = \frac{\hat{x}-r}{\tau} \tag{2.135}$$

$$\left(\Lambda^{(2)} f\right)(r,\tau) = \frac{f''(\hat{x})}{1-\tau f''(\hat{x})} \tag{2.136}$$

$$\frac{\partial}{\partial r}\hat{x} = \frac{1}{1-\tau f''(\hat{x})} \tag{2.137}$$

The idea is now to use the above properties to obtain expressions for the partial derivatives of H. In order to do that, define the *scalar function* $g_{\text{out}}(\hat{p}, y, \tau^p)$ as the partial derivative of H w.r.t. $\hat{p}$. That is,

$$g_{\text{out}}(\hat{p}, y, \tau^p) \equiv \frac{\partial}{\partial \hat{p}} H(\hat{p}, y, \tau^p) \tag{2.138}$$

Then by comparing the definition of the function H in eq. (2.121) with eq. (2.130), it is seen that

$$(Lf_{\text{out}})(z, \hat{p}, \tau^p) = f_{\text{out}}(z,y) - \frac{1}{2\tau^p}(z-\hat{p})^2 \tag{2.139}$$

and therefore eq. (2.132) implies that the function H can be written as:

$$H(\hat{p}, y, \tau^p) \approx \max_z \left\{ f_{\text{out}}(z,y) - \frac{1}{2\tau^p}(z-\hat{p})^2 \right\} = (\Lambda f_{\text{out}}(z,y))(\hat{p}, \tau^p) \tag{2.140}$$

This means that the function g_{out} can be written as:

$$g_{\text{out}}(\hat{p}, y, \tau^p) \equiv \frac{\partial}{\partial \hat{p}} H(\hat{p}, y, \tau^p) = \frac{\partial}{\partial \hat{p}} (\Lambda f_{\text{out}}(z,y))(\hat{p}, \tau^p) \tag{2.141}$$

Then by applying the definition in eq. (2.133) and the property in eq. (2.135), we get

$$g_{\text{out}}\left(\hat{p}, y, \tau^p\right) = \frac{\hat{z}^0 - \hat{p}}{\tau^p} \tag{2.142}$$

where $\hat{z}^0$ is given by

$$\hat{z}^0 = \left(\Gamma f\right)\left(\hat{p}, \tau^p\right) = \arg\max_z \left\{ f_{\text{out}}(z, y) - \frac{1}{2\tau^p}\left(z - \hat{p}\right)^2 \right\} \tag{2.143}$$

Similarly, by using the result in eq. (2.136), we get an expression for the partial derivative of $g_{\text{out}}\left(\hat{p}, y, \tau^p\right)$ w.r.t. $\hat{p}$ as well:

$$\begin{aligned}\frac{\partial}{\partial \hat{p}} g_{\text{out}}\left(\hat{p}, y, \tau^p\right) &= \frac{\partial^2}{\partial \hat{p}^2} H(\hat{p}, y, \tau^p) \\ &= \frac{f''_{\text{out}}(\hat{z}^0, y)}{1 - \tau^p f''_{\text{out}}(\hat{z}^0, y)} \end{aligned} \tag{2.144}$$

Using the expression for g_{out} and its derivative, we can now compute the coefficients for the Taylor expansion. For that purpose, we define

$$\hat{s}_a \equiv g_{\text{out}}\left(\hat{p}_a, y_a, \tau^p_a\right) \tag{2.145}$$

$$\tau^s_a \equiv -\frac{\partial}{\partial \hat{p}} g_{\text{out}}\left(\hat{p}_a, y_a, \tau^p_a\right) \tag{2.146}$$

We now return from our detour and substitute $\hat{s}_a$ and τ^s_a into the Taylor expansion in eq. (2.129) to get:

$$\mu^k_{a\to i}\left(x_i\right) \approx \hat{s}_a A_{ai}\left(x_i - \hat{x}^k_i\right) - \frac{\tau^s_a}{2} A^2_{ai}\left(x_i - \hat{x}^k_i\right)^2$$

Expanding the parentheses and rearranging:

$$\begin{aligned}\mu^k_{a\to i}\left(x_i\right) &= \hat{s}_a A_{ai} x_i - \hat{s}_a A_{ai}\hat{x}^k_i - \frac{\tau^s_a}{2} A^2_{ai}\left(x^2_i + \left(\hat{x}^k_i\right)^2 - 2x_i\hat{x}^k_i\right) \\ &= \left(\hat{s}_a A_{ai} + \tau^s_a A^2_{ai}\hat{x}^k_i\right) x_i - \frac{\tau^s_a}{2} A^2_{ai} x^2_i - \hat{s}_a A_{ai}\hat{x}^k_i + \frac{\tau^s_a}{2} A^2_{ai}\left(\hat{x}^k_i\right)^2\end{aligned}$$

Since the terms $\hat{s}_a A_{ai}\hat{x}^k_i$ and $\frac{\tau^s_a}{2} A^2_{ai}\left(\hat{x}^k_i\right)^2$ are constant w.r.t x_i, they can be absorbed into the normalization constant:

$$\mu^k_{a\to i}\left(x_i\right) \approx \left(\hat{s}_a A_{ai} + \tau^s_a A^2_{ai}\hat{x}^k_i\right) x_i - \frac{\tau^s_a}{2} A^2_{ai} x^2_i \tag{2.147}$$

We have now managed to reduce the messages from factor node to variables nodes from being a real function on the entire real line to a simply message, which is parametrized by $\{\hat{s}_a, \tau_a^s\}$. These parameters are obtained from the scalar function g_{out} and its partial derivative.

Now, we turn our attention to the messages from variable nodes to factor nodes in order to obtain a similar simplification. In order to achieve this, we substitute the above expression in eq. (2.147) into the expression for the messages from variable nodes to factor nodes in eq. (2.110) and simplify:

$$\begin{aligned}\mu_{i\to a}^{k+1}(x_i) &\approx f_{\text{in}}(x_i, q_i) + \sum_{b\neq a}\left[\left(\hat{s}_b A_{bi} + \tau_b^s A_{bi}^2 \hat{x}_i^k\right) x_i - \frac{\tau_b^s}{2} A_{bi}^2 x_i^2\right] \\ &= f_{\text{in}}(x_i, q_i) + \sum_{b\neq a}\left(\hat{s}_b A_{bi} + \tau_b^s A_{bi}^2 \hat{x}_i^k\right) x_i - \frac{1}{2} x_i^2 \sum_{b\neq a} \tau_b^s A_{bi}^2\end{aligned}$$

Now define $\tau_{i\to a}^r$ as:

$$\frac{1}{\tau_{i\to a}^r} \equiv \sum_{b\neq a} A_{bi}^2 \tau_b^s \tag{2.148}$$

Inserting this definition yields:

$$\mu_{i\to a}^{k+1}(x_i) \approx f_{\text{in}}(x_i, q_i) + \frac{\tau_{i\to a}^r}{\tau_{i\to a}^r} \sum_{b\neq a}\left(\hat{s}_b A_{bi} + \tau_b^s A_{bi}^2 \hat{x}_i^k\right) x_i - \frac{1}{2} x_i^2 \frac{1}{\tau_{i\to a}^r} \tag{2.149}$$

Furthermore, define $\hat{r}_{i\to a}$ as:

$$\hat{r}_{i\to a} \equiv \tau_{i\to a}^r \sum_{b\neq a}\left(\hat{s}_b A_{bi} + \tau_b^s A_{bi}^2 \hat{x}_i^k\right) \tag{2.150}$$

Substituting this into eq. (2.149) leads to:

$$\mu_{i\to a}^{k+1}(x_i) \approx f_{\text{in}}(x_i, q_i) + \frac{1}{\tau_{i\to a}^r}\left(\hat{r}_{i\to a} x_i - \frac{1}{2} x_i^2\right) \tag{2.151}$$

We now rewrite the term in the parenthesis as follows:

$$\begin{aligned}\left(\hat{r}_{i\to a} x_i - \frac{1}{2} x_i^2\right) &= -\frac{1}{2}\left(x_i^2 - 2\hat{r}_{i\to a} x_i\right) \\ &= -\frac{1}{2}\left(\left(\hat{r}_{i\to a} - x_i\right)^2 - \hat{r}_{i\to a}^2\right) \\ &= -\frac{1}{2}\left(\hat{r}_{i\to a} - x_i\right)^2 + k_1,\end{aligned} \tag{2.152}$$

where it is used that the term $\frac{1}{2}\hat{r}_{i\to a}^2$ is constant w.r.t. x_i. Next, substituting the result from eq. (2.152) back into eq. (2.151) gives:

$$\mu_{i\to a}^{k+1}(x_i) \approx f_{\text{in}}(x_i, q_i) - \frac{1}{2\tau_{i\to a}^r}\left(\hat{r}_{i\to a} - x_i\right)^2 \tag{2.153}$$

where the constant k_1 have been absorbed into the normalization constant.

The messages from variable nodes to factor nodes have now been considerably simplified as well and we are now ready to define the second of the two scalar functions, i.e. g_{in}:

$$g_{\text{in}}\left(\hat{r}, q, \tau^r\right) \equiv \arg\max_x \left\{ f_{\text{in}}\left(x_i, q_i\right) - \frac{1}{2\tau^r_{i\to a}} \left(\hat{r}_{i\to a} - x_i\right)^2 \right\} \tag{2.154}$$

By recalling the definition of $\hat{x}^k_{j\to a}$ in eq. (2.111), it is seen that:

$$\hat{x}_{i\to a} \equiv \arg\max_{x_i} \mu_{i\to a}(x_i) = g_{\text{in}}\left(\hat{r}_{i\to a}, q_i, \tau^r_{i\to a}\right) \tag{2.155}$$

The quantities $\hat{r}_{i\to a}$ and $\tau^r_{i\to a}$ are now approximated in analogy to the approximations of the parameters $p_{a\to i}$ and $\tau_{a\to i}$ earlier. First we make the following definitions:

$$\tau^r_i = \left[\sum_a A^2_{ai}\tau^s_a\right]^{-1}, \qquad \hat{r}_i = \hat{x}_i + \tau^r_i \sum_a A_{ai}\hat{s}_a \tag{2.156}$$

Note, that these quantities are independent of index a. We can now approximate $\tau^r_{i\to a}$ (defined in eq. (2.148)) using these definitions:

$$\tau^r_{i\to a} = \left[\sum_{b\neq a} A^2_{bi}\tau^s_b\right]^{-1} \approx \left[\sum_b A^2_{bi}\tau^s_b\right]^{-1} \equiv \tau^r_i \tag{2.157}$$

That is, we ignore the term $A^2_{ai}\tau^s_a$, which is of order $\mathcal{O}(A^2_{ai})$ and hence, this approximation is also expected to become negligible, when the system size increase. Consider now the expression for $\hat{r}_{i\to a}$. Using a number of the previous results, the expression for $\hat{r}_{i\to a}$ can be rewritten as follows:

$$\begin{aligned}
\hat{r}_{i\to a} &= \tau^r_{i\to a}\sum_{b\neq a}\left(\hat{s}_b A_{bi} + \tau^s_b A^2_{bi}\hat{x}_i\right) && \text{(Using def. (2.150))}\\
&= \tau^r_{i\to a}\sum_{b\neq a}\hat{s}_b A_{bi} + \tau^r_{i\to a}\sum_{b\neq a}\tau^s_b A^2_{bi}\hat{x}_i\\
&= \tau^r_{i\to a}\sum_{b\neq a}\hat{s}_b A_{bi} + \hat{x}_i && \text{(Using eq. (2.148))}\\
&= \tau^r_i\sum_{b\neq a}\hat{s}_b A_{bi} + \hat{x}_i && \text{(Using eq. (2.157))}\\
&= \tau^r_i\sum_{b\neq a}\hat{s}_b A_{bi} + \hat{r}_i - \tau^r_i\sum_a A_{ai}\hat{s}_a && \text{(Using eq. (2.156))}\\
&= \hat{r}_i - \tau^r_i A_{ai}\hat{s}_a
\end{aligned} \tag{2.158}$$

Substituting the approximations for $\hat{r}_{i\to a}$ and $\tau^r_{i\to a}$ back into the update equation yields:

$$\mu^{k+1}_{i\to a}(x_i) \approx f_{\text{in}}(x_i, q_i) - \frac{1}{2\tau^r_i}\left(\hat{r}_i - \tau^r_i A_{ai}\hat{s}_a - x_i\right)^2 \tag{2.159}$$

We also substitute the approximations for $\hat{r}_{i\to a}$ and $\tau^r_{i\to a}$ into the expression for $\hat{x}_{i\to a}$ in eq. (2.155) to get:

$$\begin{aligned}\hat{x}_{i\to a} &= g_{\text{in}}\left(\hat{r}_{i\to a},\, q_i,\, \tau^r_{i\to a}\right)\\ &= g_{\text{in}}\left(\hat{r}_i - \tau^r_i A_{ai}\hat{s}_a,\, q_i,\, \tau^r_i\right)\end{aligned} \tag{2.160}$$

The function $g_{\text{in}}\left(\hat{r}_i - \tau^r_i A_{ai}\hat{s}_a, q_i, \tau^r_i\right)$ is now approximated using a first order Taylor expansion around the point $\hat{r}_i$:

$$\begin{aligned}\hat{x}_{i\to a} &= g_{\text{in}}\left(\hat{r}_i, q_i, \tau^r_i\right) + \frac{\partial}{\partial\hat{r}}g_{\text{in}}\left(\hat{r}, q_i, \tau^r_i\right)\big|_{\hat{r}=\hat{r_j}}\left(\hat{r}_i - \tau^r_i A_{ai}\hat{s}_a - \hat{r}_i\right)\\ &= g_{\text{in}}\left(\hat{r}_i, q_i, \tau^r_i\right) - A_{ai}\hat{s}_a\tau^r_i\frac{\partial}{\partial\hat{r}}g_{\text{in}}\left(\hat{r}, q_i, \tau^r_i\right)\big|_{\hat{r}=\hat{r_j}}\end{aligned} \tag{2.161}$$

Based on this approximation, we will now introduce the last two definitions needed to finish this derivation. Similar to the definition of $\hat{x}_{i\to a}$ in eq. (2.155), define $\hat{x}_i$ and D_i as:

$$\hat{x}_i \equiv g_{\text{in}}\left(\hat{r}_i, q_i, \tau^r_i\right) \tag{2.162}$$

$$D_i \equiv \tau^r_i\frac{\partial}{\partial\hat{r}}g_{\text{in}}\left(\hat{r}, q_i, \tau^r_i\right)\big|_{\hat{r}=\hat{r_j}} \tag{2.163}$$

Substituting $\hat{x}_i$ and D_i into the first order approximation in eq. (2.161) gives rise to:

$$\hat{x}_{i\to a} = \hat{x}_i - A_{ai}\hat{s}_a D_i \tag{2.164}$$

The expression for D_i is now simplified as follows:

$$\begin{aligned}D_i &= \tau^r_i\frac{\partial}{\partial\hat{r}}\left(\Gamma f_{\text{in}}\right)\left(\hat{r}_i, \tau^r_i\right) && \text{Using eq. (2.132)}\\ &= \tau^r_i\frac{\partial}{\partial\hat{r}}\hat{x}_i && \text{Using def. (2.134)}\\ &= \tau^r_i\frac{1}{1 - \tau^r_i f''_{\text{in}}\left(\hat{x}_i, q_i\right)} && \text{Using eq. (2.137)}\end{aligned} \tag{2.165}$$

We will now show that $\frac{1}{1-\tau^r_i f''_{\text{in}}(\hat{x}_i)}$ is related to the second order partial derivative of $\mu_{i\to a}(x_i)$ evaluated at $\hat{x}_i$. Taking the second order partial derivative of eq.

(2.153) w.r.t. x_i yields:

$$\begin{aligned}
\frac{\partial^2}{\partial x_i^2}\mu_{i\to a}^{k+1}(x_i) &= \frac{\partial}{\partial x_i}\left[f'_{\text{in}}(x_i, q_i) + 2\frac{1}{2\tau_{i\to a}^r}(\hat{r}_{i\to a} - x_i)\right] \\
&= f''_{\text{in}}(x_i, q_i) - \frac{1}{\tau_{i\to a}^r} \\
&= \frac{\tau_{i\to a}^r f''_{\text{in}}(x_i, q_i) - 1}{\tau_{i\to a}^r} \\
&= -\left[\frac{\tau_{i\to a}^r}{1 - \tau_{i\to a}^r f''_{\text{in}}(x_i, q_i)}\right]^{-1}
\end{aligned} \tag{2.166}$$

Now by comparing eq. (2.165) and eq. (2.166), it is seen that D_i can be written as:

$$D_i = -\left[\frac{\partial^2}{\partial x_i^2}\mu_{i\to a}^{k+1}(\hat{x}_i)\right]^{-1},$$

which we in turn approximate using eq. (2.113):

$$D_i \approx \tau_i^x \tag{2.167}$$

Now we substitute eq. 2.167 back into eq. (2.164) to get:

$$\hat{x}_{i\to a} = \hat{x}_i - A_{ai}\hat{s}_a\tau_i^x \tag{2.168}$$

At last, we need to obtain update expressions for the τ^x and $\hat{p}_a$ parameters. By substituting the result from eq. (2.168) into eq. (2.123), we get the following expression for $\hat{p}_a$:

$$\begin{aligned}
\hat{p}_a &= \sum_i A_{ai}(\hat{x}_i - A_{ai}\hat{s}_a\tau_i^x) \\
&= \sum_i A_{ai}\hat{x}_i - \hat{s}_a\sum_i A_{ai}^2\tau_i^x \\
&= \sum_i A_{ai}\hat{x}_i - \hat{s}_a\tau_a^p \qquad \text{(Using def. (2.123))},
\end{aligned} \tag{2.169}$$

which is the final update equation for $\hat{p}_a$. To get to update equation for τ_i^x, we combine the definition of D_i in eq. (2.163) with the approximation in eq. (2.167) to give:

$$\tau_i^x \approx \tau_i^r \frac{\partial}{\partial \hat{r}_i} g_{\text{in}}(\hat{r}, q_i, \tau_i^r) \tag{2.170}$$

This step ends the derivation of GAMP for MAP estimation.

By means of a series of approximations, the update equations from variable nodes to factor node and vice versa were simplified to a set of parametrized messages given by:

$$\mu_{i\to a}^{k+1}(x_i) \approx f_{\text{in}}(x_i, q_i) - \frac{1}{2\tau_i^r}\left(\hat{r}_i - \tau_i^r A_{ai}\hat{s}_a - x_i\right)^2$$
$$\mu_{a\to i}^{k}(x_i) \approx \left(\hat{s}_a A_{ai} + \tau_a^s A_{ai}^2 \hat{x}_i^k\right) x_i - \frac{\tau_a^s}{2} A_{ai}^2 x_i^2,$$

where the parameters of these messages are $\tau_i^r, \hat{r}_i, \hat{s}_a, \tau_a^s$, and $\hat{x}_i$. Furthermore, the parameters are computed using the two scalar functions g_{in} and g_{out}, which are determined from the prior and noise distribution, respectively. Algorithm 2 summarizes how to update the parameters.

Derivation of Sum-Product Algorithm for MMSE Estimation

The objective of this subsection is to derive the GAMP algorithm for MMSE estimation based on the sum-product algorithm. That is, we want to estimate

$$\hat{x}^{mmse} = \mathbb{E}\left[x \middle| \boldsymbol{y}, q\right]. \tag{2.171}$$

The decomposition of the joint distribution is the same as in the MAP-case and therefore the topology of the underlying factor graph does not change. Fortunately, this implies that many of the results from the MAP derivation can be reused.

Exact Sum-Product Message Passing Equations

As before, the first step is to derive the exact messages based on the factor graph in figure 2.9. We will follow the approach in [Ran10] and use logarithmic message for the sum-product algorithm as well. Messages in the non-logarithmic space will be denoted using a "hat", e.g. $\hat{\mu}$ and messages in the logarithmic space will just be denoted μ.

Starting from the left leaves, the message from factor node $p(x_i|q_i)$ to variable node x_i simplify becomes the factor function itself:

$$\hat{\mu}_{p(x_i|q_i)\to x_i}(x_i) = p(x_i|q_i) \tag{2.172}$$

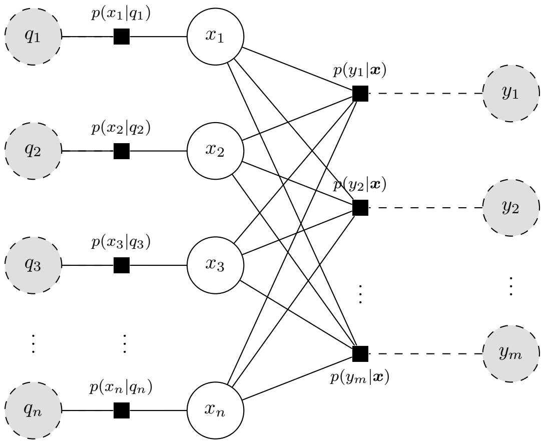


Figure 2.9: Factor graph for the joint density in eq. (2.103) for MMSE estimation

Next, the message from variable node x_i to factor node $p(y_a|\boldsymbol{x})$ is given by:

$$\begin{aligned}\hat{\mu}_{x_i \to p(y_a|\boldsymbol{x})}(x_i) &= \prod_{b \neq a} \hat{\mu}_{p(y_b|\boldsymbol{x}) \to x_i}(x_i) \hat{\mu}_{p(x_i|q_i) \to x_i}(x_i) \\ &= \prod_{b \neq a} \hat{\mu}_{p(y_b|\boldsymbol{x}) \to x_i}(x_i) p(x_i \big| q_i) \end{aligned} \tag{2.173}$$

Transforming the messages to the logarithmic-space yields:

$$\begin{aligned}\mu_{x_i \to p(y_a|\boldsymbol{x})}(x_i) &= \ln \hat{\mu}_{x_i \to p(y_a|\boldsymbol{x})}(x_i) \\ &= \sum_{b \neq a} \ln \hat{\mu}_{p(y_b|\boldsymbol{x}) \to x_i}(x_i) + \ln p(x_i \big| q_i) \\ &= \sum_{b \neq a} \mu_{p(y_b|\boldsymbol{x}) \to x_i}(x_i) + f_{\text{in},i}(x_i, q_i) \end{aligned} \tag{2.174}$$

Note, that this message is identical to the corresponding message in the max-sum GAMP algorithm. Consider now the messages in the other direction. The message from factor node $p(y_a \big| z_a)$ to variable node x_i becomes:

$$\begin{aligned}\hat{\mu}_{p(y_a|z_a) \to x_i}(x_i) &= \int p(y_a \big| z_a) \hat{\mu}_{y_a \to p(y_a|z_a)}(y_a) \prod_{j \neq i} \hat{\mu}_{x_j \to p(y_a|z_a)}(x_j) \, \mathrm{d}x_{j \neq i} \\ &= \int p(y_a \big| z_a) \prod_{j \neq i} \hat{\mu}_{x_j \to p(y_a|z_a)}(x_j) \, \mathrm{d}x_{j \neq i} \end{aligned} \tag{2.175}$$

where $z_a = \sum_i A_{ai} x_i$. The message in eq. (2.175) is equivalent to the expectation of $p(y_a \big| z_a)$ over the variable z_a with x_i being independently distributed according to $p_{i\to a}(x_i) \propto \hat{\mu}_{x_i \to p(y_a|\boldsymbol{x})}$. That is,

$$\hat{\mu}_{p(y_a|\boldsymbol{x})\to x_i}(x_i) = \mathbb{E}\left[p(y_a \middle| z_a)\right] \tag{2.176}$$

Transforming the message to the logarithmic-space yields:

$$\begin{aligned} \mu_{p(y_a|\boldsymbol{x})\to x_i}(x_i) &= \ln \hat{\mu}_{p(y_a|\boldsymbol{x})\to x_i}(x_i) \\ &= \ln \mathbb{E}\left[p(y_a \middle| z_a)\right] \end{aligned} \tag{2.177}$$

Thus, the two exact messages are given by:

$$\mu_{a\to i}(x_i) = \ln \mathbb{E}\left[p(y_a \middle| z_a)\right] \tag{2.178}$$

$$\mu_{i\to a}(x_i) = \sum_{b\neq a} \mu_{b\to x_i}(x_i) + \ln p(x_i | q_i) \tag{2.179}$$

We can now write the final estimate of the marginal posterior distribution of x_i as

$$p(x_j) \propto \exp\left[\mu_i(x_i)\right], \tag{2.180}$$

where $\mu_i(x_i)$ is defined as:

$$\mu_i(x_i) \equiv f_{\text{in},i}(x_i, q_i) + \sum_i \mu_{a\to i}(x_i) \tag{2.181}$$

The objective of the remainder of this section is to show how the two scalar functions g_{in} and g_{out} must be defined in order to perform MMSE estimation using GAMP algorithm.

Approximation the Sum-Product Messages

We will now apply a series of approximations to simplify the message passing equations from above. In order to do that, we will need the following definitions:

$$\hat{x}_i \equiv \mathbb{E}\left[x_i \middle| \mu_i(\cdot)\right] \tag{2.182}$$

$$\hat{x}_{i\to a} \equiv \mathbb{E}\left[x_i \middle| \mu_{i\to a}(\cdot)\right] \tag{2.183}$$

$$\tau_i^x \equiv \text{var}\left[x_i \middle| \mu_i(\cdot)\right] \tag{2.184}$$

$$\tau_{i\to a}^x \equiv \text{var}\left[x_i \middle| \mu_{i\to a}(\cdot)\right] \tag{2.185}$$

where $\mathbb{E}\left[g(x) \middle| \mu(\cdot)\right]$ means the expectation over x with density $p(x) \propto \exp\left[\mu(x)\right]$. This means that $\hat{x}_{i\to a}$ and $\tau_{i\to a}^x$ are the mean and variance, respectively, of a random variable x_i with density $\propto \exp\left[\mu_{i\to a}\right]$.

Consider the messages from factor nodes to variable nodes, i.e. eq. (2.178). In this equation, the expectation is over $z_a = \sum_j A_{ai} x_j$ and therefore, for large n the central limit theorem suggests that z_a conditioned on x_i is approximately Gaussian distributed with mean and variance given by:

$$\mathbb{E}\left[z_a\right] = A_{ai} x_i + \sum_{j \neq i} A_{aj} \mathbb{E}\left[x_j \middle| \mu_{i \to a}\left(\cdot\right)\right] \tag{2.186}$$

$$\mathbb{V}\left[z_a\right] = \sum_{j \neq i} A_{aj}^2 \mathbb{V}\left[x_j \middle| \mu_{i \to a}\left(\cdot\right)\right] \tag{2.187}$$

where the variance of x_i is zero since we are conditioning on x_i. Therefore, we have that

$$z_a \big| x_i \sim \mathcal{N}(A_{ai} x_i + \sum_{j \neq i} A_{aj} \mathbb{E}\left[x_j \middle| \mu_{i \to a}\left(\cdot\right)\right] \; , \sum_{j \neq i} A_{aj}^2 \; \mathbb{V}\left[x_j \middle| \mu_{i \to a}\left(\cdot\right)\right]) \tag{2.188}$$

Using the definitions in eq. (2.183) and eq. (2.185) from above, this can be rewritten as

$$z_a \big| x_i \sim \mathcal{N}(A_{ai} x_i + \sum_{j \neq i} A_{aj} \hat{x}_{i \to a}, \sum_{j \neq i} A_{aj}^2 \tau_{i \to a}^x) \tag{2.189}$$

Now recognize that we can use the definitions of $\hat{p}_{a \to i}$ and $\tau_{a \to i}^p$ in eq. (2.118) from earlier to write:

$$z_a \big| x_i \sim \mathcal{N}\left(z_a \,\middle|\, \hat{p}_{a \to i} + A_{ai} x_i, \tau_{a \to i}^p\right) \tag{2.190}$$

This implies that the messages from factor nodes to variables nodes can be written as:

$$\begin{aligned} \mu_{a \to i}(x_i) &= \ln \mathbb{E}\left[p(y_a \big| z_a)\right] \\ &\approx \ln \int p(y_a \big| z_a) \mathcal{N}\left(z_a \big| \hat{p}_{a \to i} + A_{ai} x_i, \tau_{a \to i}^p\right) \, \mathrm{d} z_a \end{aligned}$$

Analogous to the derivation of the MAP algorithm, we now define the function H as follows[7]:

$$H(\hat{p}, y, \tau^p) \equiv \ln \mathbb{E}\left[p(y \big| z)\right] \tag{2.191}$$

where the expectation is over $z \sim \mathcal{N}\left(z \,\middle|\, \hat{p}, \tau^p\right)$. The message $\mu_{a \to i}(x_i)$ can now be rewritten using H:

$$\mu_{a \to i}(x_i) \approx H(\hat{p}_{a \to i} + A_{ai} x_i, y_a, \tau_{a \to i}^p) \tag{2.192}$$

We now recognize that the result above, i.e. eq. (2.192) has the exact same form as eq. (2.122) from the MAP derivation. This means that we can apply

[7]Note, the expectation operator is missing in eq. (130) in the original paper [Ran10]

the exact same approximations as we did in the MAP-case. Therefore, we can approximate $\mu_{a\to i}(x_i)$ using the result in eq. (2.127). This yields:

$$\mu_{a\to i}(x_i) \approx H\left(\hat{p}_a + A_{ai}\left(x_i - \hat{x}_i\right),\, y_a,\, \tau_a^p\right) \tag{2.193}$$

Again, completely analogous to the MAP case, a second order approximation of eq. (2.193) yields:

$$\mu_{a\to i}(x_i) \approx \hat{s}_a A_{ai}\left(x_i - \hat{x}_i\right) - \frac{1}{2}\tau_a^s\left(A_{ai}\left(x_i - \hat{x}_i\right)\right)^2, \tag{2.194}$$

where $g_{out}(\hat{p}, y, \tau^p)$, $\hat{s}_a$ and τ_a^s are defined as in eq. (2.138), (2.145), and (2.146), respectively.

The expression for the scalar function g_{out} and its partial derivative $\frac{\partial}{\partial\hat{p}}g_{\text{out}}$ are now computed. Using the definition of H in eq. (2.191), we get:

$$\begin{aligned} g_{\text{out}}(\hat{p}, y, \tau^p) &= \frac{\partial}{\partial\hat{p}} H(\hat{p}, y, \tau^p) \\ &= \frac{1}{\mathbb{E}\left[p(y|z)\right]}\frac{\partial}{\partial\hat{p}}\mathbb{E}\left[p(y|z)\right] \end{aligned}$$

To ease the notation, let Z denote the normalization constant, i.e. $Z = \mathbb{E}\left[p(y|z)\right]$:

$$\begin{aligned} g_{\text{out}}(\hat{p}, y, \tau^p) &= \frac{\partial}{\partial\hat{p}}\mathbb{E}\left[p(y|z)\right] \\ &= \frac{1}{Z}\int p(y|z)\frac{\partial}{\partial\hat{p}}\mathcal{N}\left(z|\hat{p}, \tau^p\right)\,\mathrm{d}z \\ &= \frac{1}{Z}\int p(y|z)\frac{z-\hat{p}}{\tau^p}\mathcal{N}\left(z|\hat{p}, \tau^p\right)\,\mathrm{d}z \\ &= \frac{1}{Z}\frac{1}{\tau^p}\int zp(y|z)\mathcal{N}\left(z|\hat{p}, \tau^p\right)\,\mathrm{d}z - \frac{\hat{p}}{\tau^p}\frac{1}{Z}\int p(y|z)\mathcal{N}\left(z|\hat{p}, \tau^p\right)\,\mathrm{d}z \\ &= \frac{1}{\tau^p}\left(\hat{z}^0 - \hat{p}\right), \end{aligned} \tag{2.195}$$

where

$$\hat{z}^0 = \mathbb{E}\left[z|\hat{p}, \tau^p\right] = \frac{\int zp(y|z)\mathcal{N}\left(z|\hat{p}, \tau^p\right)\,\mathrm{d}z}{\int p(y|z)\mathcal{N}\left(z|\hat{p}, \tau^p\right)\,\mathrm{d}z} \tag{2.196}$$

For the partial derivative of g_{out} w.r.t. $\hat{p}$, we get:

$$\frac{\partial}{\partial\hat{p}}g_{\text{out}}(\hat{p}, y, \tau^p) = \frac{1}{\tau^p}\left(\frac{\partial}{\partial\hat{p}}\hat{z}^0 - 1\right)$$

Thus, we need to compute the partial derivative of $\hat{z}^0$, which is given by:

$$\frac{\partial}{\partial \hat{p}} \hat{z}^0 = \frac{\int z p(y|z) \mathcal{N}\left(z|\hat{p}, \tau^p\right) \, \mathrm{d}z}{\int p(y|z) \mathcal{N}\left(z|\hat{p}, \tau^p\right) \, \mathrm{d}z} \tag{2.197}$$

Applying the quotient rule for derivatives and using the fact that we can exchange the order of the derivative and the integration gives:

$$\frac{\partial}{\partial \hat{p}} \hat{z}^0 = \frac{1}{\tau^p} \mathbb{V}\left[z|\hat{p}, \tau^p\right], \tag{2.198}$$

where the scaling $1/\tau^p$ comes from the computing the partial derivative of the Gaussian density:

$$\frac{\partial}{\partial \hat{p}} \mathcal{N}\left(z|\hat{p}, \tau^p\right) = \frac{z - \hat{p}}{\tau^p} \mathcal{N}\left(z\big|\hat{p}, \tau^p\right)$$

Therefore, the partial derivative of the scalar function g_{out} w.r.t $\hat{p}$ becomes:

$$\frac{\partial}{\partial \hat{p}} g_{\text{out}}(\hat{p}, y, \tau^p) = \frac{1}{\tau^p} \left(\frac{1}{\tau^p} \mathbb{V}\left[z|\hat{p}, \tau^p\right] - 1 \right), \tag{2.199}$$

Consider now the message update in the other direction. Inserting the approximation of the messages $\mu_{a \to i}(x_i)$ from eq. (2.194) into the expression $\mu_{i \to a}(x_i)$ in eq.(2.179) yields:

$$\begin{aligned}
\mu_{i \to a}(x_i) &= \sum_{b \neq a} \mu_{b \to i}(x_i) + \ln p(x_i \big| q_i) \\
&\approx \sum_{b \neq a} \left[\left(A_{bi} \hat{s}_b - A_{bi}^2 \tau_b^s \hat{x}_i \right) x_i - \frac{\tau_b^s}{2} A_{bi}^2 x_i^2 \right] + \ln p(x_i \big| q_i)
\end{aligned}$$

We can now repeat the exact same calculation as in the MAP case to obtain:

$$\mu_{i \to a}^k (x_i) \approx f_{in}(x_i, q_i) - \frac{1}{2\tau_{i \to a}^r} \left(\hat{r}_{i \to a} - x_i \right)^2, \tag{2.200}$$

where $\hat{r}_{i \to a}$ and $\tau_{i \to a}^r$ are defined in eq. (2.150) and (2.148), respectively. Finally, using the fact that $\mu_{i \to a}(x_i) = \ln \hat{\mu}_{i \to a}(x_i)$, we can write:

$$\begin{aligned}
\hat{\mu}_{i \to a}(x_i) &\approx \frac{1}{Z} \exp\left[\mu_{i \to a}(x_i)\right] \\
&= \frac{1}{Z} \exp\left[f_{in}(x_i, q_i) - \frac{1}{2\tau_{i \to a}^r} \left(\hat{r}_{i \to a} - x_i \right)^2 \right]
\end{aligned} \tag{2.201}$$

where Z is a normalizing constant.

We are now ready to define the second scalar function, g_{in}, as:

$$g_{\text{in}}(\hat{r}, q, \tau^r) \equiv \mathbb{E}\left[x_i \middle| \mu_{i\to a}(\cdot)\right] \tag{2.202}$$

where the expectation is over the density in eq. (2.201). Then by using definition in eq. (2.183) and the result above, we have that:

$$\hat{x}_{i\to a} \equiv \mathbb{E}\left[x_i \middle| \mu_{i\to a}(\cdot)\right] = g_{\text{in}}(\hat{r}_{i\to a}, q_i, \tau^r_{i\to a}) \tag{2.203}$$

Note, that this result has exactly the same form as the corresponding result for the MAP-case given in eq. (2.160) and therefore we can apply the same approximations to obtain the expression in eq. (2.162). The result is restated here for convenience:

$$\begin{aligned}\hat{x}_i &\equiv g_{\text{in}}\left(\hat{r}_i, q_i, \tau^r_i\right), \\ &= \mathbb{E}\left[x_i \middle| \hat{r}_i, q_i, \tau^r_i\right]\end{aligned} \tag{2.204}$$

where $\hat{r}_i$ and τ^r_i are given in eq. (2.156) and eq. (2.157), respectively. Using a similar line of arguments as for the calculation of the partial derivative of g_{out} in eq. (2.198), the partial derivative of g_{in} w.r.t. $\hat{r}$, is given by:

$$\frac{\partial}{\partial \hat{r}} g_{\text{in}}\left(\hat{r}_i, q_i, \tau^r_i\right) = \frac{1}{\tau^r}\mathbb{V}\left[x_i|\hat{r}, \tau^r\right], \tag{2.205}$$

Thus, explicit expressions for the two scalar functions have been derived. This finalizes the derivation of the GAMP algorithm for MMSE estimation. In the next section, we will use the GAMP algorithm to derive an inference algorithm for MMSE estimation.

2.4 Inference using a Bernoulli-Gaussian Prior

The goal of this section is to use the GAMP1 algorithm to derive an algorithm for solving the linear inverse problem using a Bernoulli-Gaussian (BG) prior. The BG prior, which is also known as the "spike and slab" prior, is given by:

$$p(x_i|\boldsymbol{q}) = (1-\lambda)\,\delta\left(x_i\right) + \lambda\mathcal{N}\left(x_i \middle| \zeta, \psi\right) \tag{2.206}$$

where the $\lambda \in [0,1]$ is controlling the level of sparsity and $\zeta, \psi \in \mathbb{R}$ are mean and variance of the Gaussian component, respectively. Note, that the sparsity parameter λ should not be confused with the sparsity $\rho = \frac{k}{m}$ defined in section 2.2. However, λ can be interpreted as the expected number of non-zero elements in the solution. Using this interpretation, we can write:

$$\lambda = \frac{k}{n} = \frac{k}{m}\frac{m}{n} = \rho \cdot \delta \tag{2.207}$$

This particular kind of prior is also known as a *sparsity promoting prior* due to the fact that for $\lambda < 1$, the resulting density has point mass at $x_i = 0$, and hence favours sparsity. In [VS13], Vila et al. introduces an extension of the spike and slap prior, namely:

$$p_{\text{GMM}}(x_i|\boldsymbol{q}) = (1-\lambda)\,\delta\left(x_i\right) + \lambda \sum_{\ell=1}^{L} \omega \mathcal{N}\left(x_i|\zeta_\ell, \psi_\ell\right) \tag{2.208}$$

That is, the distribution of the *active coefficients* is a Gaussian Mixture Model [Bis06] rather than a single Gaussian component. They argue that this approach yields better performance in the cases, where the true prior distribution is more complex than a simple Gaussian. However, for simplicity the approach taken here is restricted to one Gaussian component only, i.e. the conventional spike and slap prior, but we note that the extension to a Gaussian mixture model is possible and a natural extension.

Another representation of the Bernoulli Gaussian distribution is that X is composed as a product of two hidden random variables, a binary support variable $S \sim \text{Ber}(\lambda)$, which is Bernoulli distributed with parameter λ and an amplitude or coefficient variable $\theta \sim \mathcal{N}\left(\zeta, \psi\right)$, which is Gaussian distributed with mean ζ and variance ψ. That is,

$$x_i = s_i \theta_i \qquad \Longleftrightarrow \qquad p(x_i|s_i, \theta_i) = \begin{cases} \delta\left(x_i\right) & \text{if } s_i = 0 \\ \delta\left(x_i - \theta_i\right) & \text{if } s_i = 1 \end{cases} \tag{2.209}$$

This representation will prove useful, when the model is extended to the multiple measurement vector problem in the next section.

The specific values of the hyperparameters for the spike and slap prior have great impact on the resulting performance of the algorithm. Therefore, following the work in [VS13], we will derive an Expectation-Maximization scheme [DLR11] to learn these hyperparameters from the data at hand. This algorithm is therefore referred to as *EMBGAMP*. From the view point of GAMP, the hyperparameter will be considered as known and fixed. Hence, this method can be classified as an Emperical-Bayes approach.

The noise is assumed to be independent, zero-mean, and Gaussian distributed with variance σ^2. Thus, the set of hyperparameters for this model then becomes $\boldsymbol{q} = \left[\lambda, \zeta, \psi, \sigma^2\right]$. Using GAMP in MMSE mode, the approximated posterior distribution of x_i is obtain using eq. (2.95):

$$p(x_i|\boldsymbol{y}, \boldsymbol{q}) \equiv \frac{1}{Z} p_x(x_i|\boldsymbol{q}) \mathcal{N}\left(x_i\middle|\,\hat{r}_i, \tau_i^r\right) \tag{2.210}$$

where Z is the normalization term given by

$$Z = \int p_x\left(x_i \middle| \boldsymbol{q}\right) \mathcal{N}\left(x_i \middle| \hat{r}_i, \tau_i^r\right) \mathrm{d}x_i \tag{2.211}$$

Now by plugging eq. (2.206) into eq. (2.210) and simplifying, we obtain

$$\begin{aligned} p(x_i|\boldsymbol{y}, \boldsymbol{q}) &= \frac{1}{Z}\left(\left(1-\lambda\right)\delta(x_i) + \lambda\mathcal{N}\left(x_i|\zeta,\psi\right)\right)\mathcal{N}\left(x_i\middle|\hat{r}_i,\tau_i^r\right) \\ &= \frac{1}{Z}\left(1-\lambda\right)\delta(x_i)\mathcal{N}\left(x_i\middle|\hat{r}_i,\tau_i^r\right) + \frac{1}{Z}\lambda\mathcal{N}\left(x_i|\zeta,\psi\right)\mathcal{N}\left(x_i\middle|\hat{r}_i,\tau_i^r\right) \end{aligned}$$

Invoking the Gaussian multiplication rule (see Appendix A.1) yields:

$$\begin{aligned} p_{x|\boldsymbol{y}}(x_i|\boldsymbol{y}, \boldsymbol{q}) = &\frac{1}{Z}\left(1-\lambda\right)\delta(x_i)\mathcal{N}\left(x_i\middle|\hat{r}_i,\tau_i^r\right) \\ &+ \frac{1}{Z}\lambda\mathcal{N}\left(x\middle|\frac{\frac{\zeta}{\psi}+\frac{\hat{r}_i}{\tau_i^r}}{\frac{1}{\psi}+\frac{1}{\tau_i^r}}, \frac{1}{\frac{1}{\psi}+\frac{1}{\tau_i^r}}\right)\mathcal{N}\left(0\middle|\zeta+\hat{r}_i,\psi+\tau_i^r\right) \end{aligned}$$

It is seen that the approximate posterior density also has the form of a spike and slap density. We now introduce the following definitions for the posterior mean and variance of the "slap"-part of the posterior density for x_i:

$$\gamma_i = \frac{\frac{\zeta}{\psi}+\frac{\hat{r}_i}{\tau_i^r}}{\frac{1}{\psi}+\frac{1}{\tau_i^r}} \qquad\qquad \nu_i = \frac{1}{\frac{1}{\psi}+\frac{1}{\tau_i^r}} \tag{2.212}$$

Similarly, we define π_i to be the posterior probability of the i'th coefficient being active, i.e. $\pi_i = p(s_i = 1\middle|\boldsymbol{y}, \boldsymbol{q})$. Substituting in these definitions gives:

$$p_{x|\boldsymbol{y}}(x_i|\boldsymbol{y}, \boldsymbol{q}) = \left(1-\pi_i\right)\delta(x_i) + \pi_i\mathcal{N}\left(x_i|\gamma_i,\nu_i\right) \tag{2.213}$$

To obtain an expression for π_i, we first compute the normalization factor Z. The calculation is tedious, but straightforward:

$$\begin{aligned} Z &= \int \left(1-\lambda\right)\delta(x_i)\mathcal{N}\left(x_i\middle|\hat{r}_i,\tau_i^r\right) + \lambda\mathcal{N}\left(x_i|\zeta,\psi\right)\mathcal{N}\left(x_i\middle|\hat{r}_i,\tau_i^r\right)\mathrm{d}x_i \\ &= (1-\lambda)\mathcal{N}\left(0\middle|\hat{r}_i,\tau_i^r\right) + \lambda\mathcal{N}\left(0\middle|\zeta-\hat{r}_i,\psi+\tau_i^r\right)\int\mathcal{N}\left(x_i\middle|\frac{\frac{\zeta}{\psi}+\frac{\hat{r}_i}{\tau_i^r}}{\frac{1}{\psi}+\frac{1}{\tau_i^r}}, \frac{1}{\frac{1}{\psi}+\frac{1}{\tau_i^r}}\right)\mathrm{d}x_i \\ &= (1-\lambda)\mathcal{N}\left(0\middle|\hat{r}_i,\tau_i^r\right) + \lambda\mathcal{N}\left(0\middle|\zeta-\hat{r}_i,\psi+\tau_i^r\right) \end{aligned} \tag{2.214}$$

We can now substitute the normalization Z into the expression for the marginal posterior distribution of x_i as given in eq. (2.210):

$$p_{x|\boldsymbol{y}}(x_i|\boldsymbol{y}, \boldsymbol{q}) = \frac{\left(1-\lambda\right)\delta(x_i)\mathcal{N}\left(x_i\middle|\hat{r}_i,\tau_i^r\right) + \lambda\mathcal{N}\left(x|\gamma_n,\nu_n\right)}{(1-\lambda)\mathcal{N}\left(0\middle|\hat{r}_i,\tau_i^r\right) + \lambda\mathcal{N}\left(0\middle|\zeta-\hat{r}_i,\psi+\tau_i^r\right)} \tag{2.215}$$

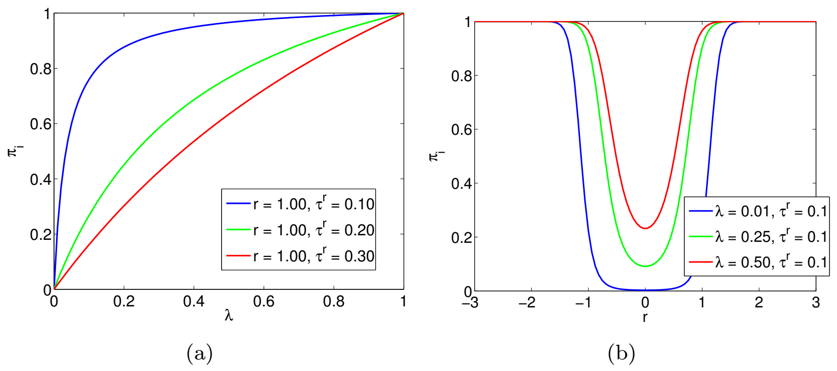


Figure 2.10: Plots of the posterior activation probability π_i for $\zeta = 0, \psi = 1$ as a function of λ (left) and $\hat{r}$ (right).

Now comparing coefficients in eq. (2.213) and eq. (2.215) and solving for π_i yields:

$$1 - \pi_i = \frac{(1-\lambda)\mathcal{N}\left(0\,\middle|\,\hat{r}_i,\, \tau_i^r\right)}{(1-\lambda)\,\mathcal{N}\left(0\middle|\,\hat{r}_i, \tau_i^r\right) + \lambda\mathcal{N}\left(0\middle|\,\zeta - \hat{r}_i, \psi + \tau_i^r\right)}$$

$$\iff \qquad \pi_i = \frac{1}{1 + \frac{(1-\lambda)\mathcal{N}\left(0\middle|\,\hat{r}_i, \tau_i^r\right)}{\lambda\mathcal{N}\left(0\middle|\,\zeta - \hat{r}_i, \psi + \tau_i^r\right)}} \tag{2.216}$$

In order to investigate how this algorithm works, figure 2.10 shows two plots of the posterior activation probability π_i as a function of λ and $\hat{r}$, respectively. It is seen that the posterior activation probability behaves as one might expect intuitively. The left-most plot shows π_i as a function of λ for $\hat{r} = 1$. It is seen that the smaller value of τ^r, i.e. the less uncertainty about $\hat{x}_i$, the higher value of π_i. The right-most plot shows π_i as a function of $\hat{r}$, and it is seen that for small λ, the posterior probability of activation is also small close to $r = 0$, and increases rapidly when the magnitude of $\hat{r}$ exceeds a threshold value.

Computing the Scalar Functions

We have now established closed form expressions for the approximate posterior quantities using the GAMP approximation. The next step is to compute the two required scalar functions and their partial derivatives. Consider the input function, $g_{in}(\hat{r}, q, \tau^r)$, which is given by:

$$\begin{aligned} g_{in}(\hat{r}, q, \tau^r) &= \mathbb{E}\left[x_i | \hat{r}, \tau^r\right] \\ &= \int x_i \left[(1-\pi_i)\,\delta(x_i) + \pi_i \mathcal{N}\left(x_i \middle| \gamma_i, \nu_i\right)\right] \mathrm{d}x_i \end{aligned}$$

Applying the linearity property of integrals yields:

$$g_{in}(\hat{r}, q, \tau^r) = (1-\pi_i)\int x_i\, \delta(x_i)\, \mathrm{d}x_i + \pi_i \int x_i \mathcal{N}\left(x_i \middle| \gamma_i, \nu_i\right) \mathrm{d}x$$

Using the sift property of Dirac delta function under an integral, it is seen that the first term evaluates to zero. The integral in the second term simply evaluates to the mean of the Gaussian density, i.e. γ_i:

$$g_{in}(\hat{r}, q, \tau^r) = \pi_i \gamma_i \tag{2.217}$$

Next we need to determine the conditional variance, which is defined as:

$$\mathbb{V}\left[X | \hat{R} = \hat{r}, \hat{Q} = q\right] = \int (x_i - \pi_i\gamma_i)^2 \left[(1-\pi_i)\,\delta(x_i) + \pi_i \mathcal{N}\left(x_i \middle| \gamma_i, \nu_i\right)\right] \mathrm{d}x_i$$

Using a similar line of arguments as for the mean, we get:

$$\begin{aligned} \mathbb{V}\left[X | \hat{R} = \hat{r}, \hat{Q} = q\right] &= \int (x_i - \pi_i\gamma_i)^2 (1-\pi_i)\,\delta(x_i) + (x_i - \pi_i\gamma_i)^2 \pi_i \mathcal{N}\left(x_i \middle| \gamma_i, \nu_i\right) \mathrm{d}x_i \\ &= (1-\pi_i)\int (x_i - \pi_i\gamma_i)^2 \delta(x_i)\, \mathrm{d}x_i + \pi_i \int (x_i - \pi_i\gamma_i)^2 \mathcal{N}\left(x_i \middle| \gamma_i, \nu_i\right) \mathrm{d}x_i \\ &= \pi\left(\gamma^2 - \pi\gamma^2 + \nu\right) \end{aligned} \tag{2.218}$$

where it is used that $\int x^2 \mathcal{N}\left(x | \mu, \sigma^2\right) = \sigma^2 + \mu^2$. The partial derivative of the scalar function $g_{\text{in}}(\hat{r}, q, \tau^p)$ then becomes

$$\begin{aligned} \frac{\partial}{\partial \hat{r}} g_{\text{in}} &= \frac{1}{\tau^r} \mathbb{V}\left[x_i | \hat{r}, \tau^r\right] \\ &= \frac{1}{\tau^r} \pi_i \left(\gamma^2 - \pi\gamma^2 + \nu\right) \end{aligned} \tag{2.219}$$

Finally, we need to compute the output functions from eq. (2.195). For convenience, the definition is restated here:

$$g_{out}\left(\hat{p}, y, \tau^p\right) \equiv \frac{1}{\tau^p}\left(\hat{z}^0 - \hat{p}\right), \qquad \hat{z}^0 \equiv \mathbb{E}\left[z | \hat{p}, y, \tau^p\right] \tag{2.220}$$

The variables $\hat{p}$ and τ^p are readily available from the GAMP approximation, therefore we only have to determine an expression for $\hat{z}^0$. Since the noise is assumed to be Gaussian, the posterior of z conditioned on y is proportional to:

$$p(z | y) \propto \mathcal{N}\left(y | z, \sigma^2\right) \mathcal{N}\left(z | \hat{p}, \tau^p\right) \tag{2.221}$$

and since both factors are Gaussian, the resulting posterior distribution is also Gaussian. The moments of this posterior distribution is then easily obtained using standard formulas for Gaussian distributions [Bis06, ch. 2.3]

$$p(z|y) = \mathcal{N}\left(z|\hat{z}^0, \tau^z\right) \tag{2.222}$$

where

$$\hat{z}^0 = \tau^z \left(\frac{y}{\sigma^2} + \frac{\hat{p}}{\tau^p}\right), \qquad \tau^z = \frac{\tau^p \sigma^2}{\tau^p + \sigma^2} \tag{2.223}$$

Therefore the desired conditional expectation becomes:

$$\mathbb{E}\left[z|\hat{p}, y, \tau^p\right] = \int z\mathcal{N}\left(z|\hat{z}^0, \tau^z\right) \mathrm{d}z = \hat{z}^0 \tag{2.224}$$

We can now compute partial derivative of $g_{\text{out}}(\hat{p}, y, \tau^p)$ w.r.t. $\hat{p}$ using eq. (2.197):

$$\begin{aligned}\frac{\partial}{\partial \hat{p}} g_{\text{out}}(\hat{p}, y, \tau^p) &\equiv \frac{1}{\tau^p}\left(\frac{1}{\tau^p}\mathbb{V}\left[z|\hat{p}, \tau^p\right] - 1\right) \\ &= \frac{1}{\tau^p}\left(\frac{1}{\tau^p}\tau^z - 1\right)\end{aligned} \tag{2.225}$$

After defining the two scalar functions, we can now simply apply the GAMP algorithm in Algorithm 2. The initial values of $\hat{x}_i^1$, $\forall i \in [n]$ should be initialized as the mean of the prior. Similarly, the variances $\tau_i^x(1)$ should be initialized as the variance of the prior. The BG-AMP algorithm is summarized in Algorithm 4.

Regarding the stopping criteria, one can either stop when a fixed number of iterations is reached or when it converges in the relative difference of the norm of the estimate of $\boldsymbol{x}$, or both. By relative difference of norm is meant the following quantity:

$$\frac{||\hat{\boldsymbol{x}}^k - \hat{\boldsymbol{x}}^{k-1}||_2^2}{||\hat{\boldsymbol{x}}^k||_2^2}. \tag{2.226}$$

We have now derived an algorithm for sparse inference under the spike and slap prior based on a set of known hyperparameters. Sometimes it is possible to give some rough estimate of the hyperparameters, while other times it is not. But in either case, it is likely that the hyperparameters will benefit from some tuning. In the following we will derive a set of EM-based update equations for the hyperparameters based on the work in [VS13]. When the hyperparameters are learned using the EM scheme, the algorithm is referred to as *EMBGAMP*.

Algorithm 4 BGAMP algorithm (BGAMP)

```
• Initialize
      Set all $\hat{x}_i^1$ as the mean value of the prior
      Set all variances $\tau_i^x(1)$ as the the variance of the prior
      Set all $s_i^0 = 0$.
• repeat until stopping criteria:
      Step 1. For each $a \in [m]$:
          $z_a^k = \sum_j A_{aj}\hat{x}_j^k$
          $\tau_a^p(k) = \sum_j A_{aj}^2 \tau_j^x(k)$
          $\hat{p}_a^k = z_a^k - \tau_a^p(k)\hat{s}_a^{k-1}$
      Step 2. For each $a \in [m]$:
          $\tau_a^z(k) = \mathbb{V}\left[z_a \middle| \boldsymbol{y}, \hat{p}_a^k, \tau_a^p(k)\right]$
          $\hat{z}(k) = \mathbb{E}\left[z_a \middle| \boldsymbol{y}, \hat{p}_a^k, \tau_a^p(k)\right]$
          $\hat{s}_a^k = \frac{\hat{z}_a^k - \hat{p}_a^k}{\tau_a^p(k)}$
          $\tau_a^s(k) = \frac{1}{\tau_a^p(k)}\left(1 - \frac{\tau_a^z(k)}{\tau_a^p(k)}\right)$
      Step 3: For each $i \in [n]$:
          $\tau_i^r(k) = \left(\sum_a A_{ai}^2 \tau_a^s(k)\right)^{-1}$
          $\hat{r}_i^k = \hat{x}_i^k + \tau_i^r(k)\sum_a A_{ai}\hat{s}_a^k$
      Step 4: For each $i \in [n]$:
          $\hat{x}_i^{k+1} = \mathbb{E}\left[x_i \middle| \boldsymbol{y}, \hat{r}_i^k, \tau_i^r(k)\right]$
          $\tau_i^x(k+1) = \mathbb{V}\left[x_i \middle| \boldsymbol{y}, \hat{r}_i^k, \tau_i^r(k)\right]$
```

Learning the Hyperparameter using Expectation Maximization

The Expectation-Maximization (EM) framework [Bis06, DLR11] is an iterative method for likelihood optimization in probabilistic models with latent or hidden variables. The EM algorithm increases a lower bound on the likelihood $p(\boldsymbol{y}|\boldsymbol{q})$ in each iterations, and therefore it is guaranteed to converge to a stationary point, i.e. a local maxima or a saddle point. The EM algorithm proceed by alternately doing the so-called E-steps and M-steps. As we will see soon, the E-step corresponds to computing an expectation, which is then maximized in the M-step. Hence, the name of the algorithm.

Let $p(\boldsymbol{y}|\boldsymbol{q})$ be the likelihood given the hyperparameters $\boldsymbol{q}$. Then for any probability density function $p(\boldsymbol{x})$, the following decomposition holds

$$\begin{aligned}
\ln p(\boldsymbol{y}|\boldsymbol{q}) &= \ln p(\boldsymbol{y}|\boldsymbol{q}) \int p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \\
&= \int p(\boldsymbol{x}) \ln \left(\frac{p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{q})}{p(\boldsymbol{x})} \frac{p(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{y}|\boldsymbol{q})} \right) \mathrm{d}\boldsymbol{x} \\
&= \int p(\boldsymbol{x}) \ln \left(\frac{p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{q})}{p(\boldsymbol{x})} \right) \mathrm{d}\boldsymbol{x} + \int p(\boldsymbol{x}) \ln \left(\frac{p(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{y}|\boldsymbol{q})} \right) \mathrm{d}\boldsymbol{x} \\
&= \int p(\boldsymbol{x}) \ln \left(p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{q}) \right) \mathrm{d}\boldsymbol{x} - \int p(\boldsymbol{x}) \ln \left(p(\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} + \int p(\boldsymbol{x}) \ln \left(\frac{p(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{y}|\boldsymbol{q})} \right) \mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_p \left[\ln p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{q}) \right] + H(p(\boldsymbol{x})) + KL\left(p(\boldsymbol{x}), p(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{q}) \right)
\end{aligned}$$

where $H(p(\boldsymbol{x}))$ is recognized as the entropy, $KL(p_1(\boldsymbol{x}), p_2(\boldsymbol{x}))$ is the Kullback-Leibler (KL) divergence [Mac03] between the two distributions $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$. Then by defining

$$\mathcal{L}_p(\boldsymbol{y}; \boldsymbol{q}) = \mathbb{E}_p \left[\ln p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{q}) \right] + H(p), \tag{2.227}$$

we can write the likelihood of $\boldsymbol{y}$ given $\boldsymbol{q}$ as:

$$\ln p(\boldsymbol{y}|\boldsymbol{q}) = \mathcal{L}_p(\boldsymbol{y}; \boldsymbol{q}) + KL\left(p(\boldsymbol{x})\, p(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{q}) \right), \tag{2.228}$$

Informally, the KL divergence measures the "distance" between two probability density functions and can be considered a pseudo-metric for probability density functions, but it is not a true metric due to lack of symmetric. But for any p_1 and p_2, the KL divergence is non-negative, i.e. $KL(p_1, p_2) \geq 0$ and $KL(p_1, p_2) = 0$ if and only if $p_1(\boldsymbol{x}) = p_2(\boldsymbol{x})$.

Since the KL divergence is non-negative, we can consider $\mathcal{L}(\boldsymbol{y}; \boldsymbol{q})$ in eq. (2.228) as a lower bound on the likelihood $\ln p(\boldsymbol{y}|\boldsymbol{q})$. Thus, we can optimize the likelihood by iteratively performing the following two steps. In the first step (E-step),

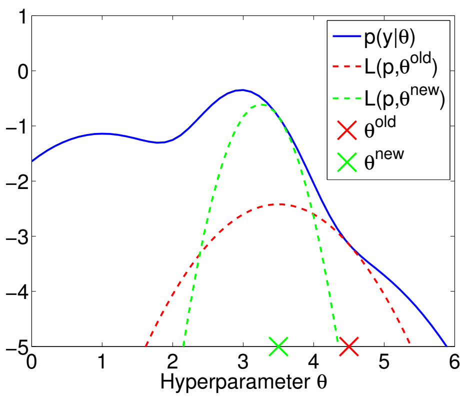


Figure 2.11: Illustration of optimization of some hyperparameter θ using EM. The blue curve is the likelihood $p(\boldsymbol{y}|\boldsymbol{q})$, θ^{old} (red cross) denotes the initial value of the hyperparameter. Then by executing the first E-step, the lower bound $\mathcal{L}_p(\boldsymbol{y}\big|\boldsymbol{q}^{old})$ (red curve) is obtained. Next, maximizing $\mathcal{L}_p(\boldsymbol{y}\big|\boldsymbol{q})$ w.r.t. $\boldsymbol{q}$ and leads to θ^{new} (green cross). Performing another E-step yields the new and improved bound $\mathcal{L}_p(\boldsymbol{y}\big|\boldsymbol{q}^{new})$ (green).

we optimize $\mathcal{L}\left(\boldsymbol{y};\boldsymbol{q}\right)$ w.r.t. $p(\boldsymbol{x})$ for a fixed $\boldsymbol{q}$ and in the second step (M-step), we maximize $\mathcal{L}\left(\boldsymbol{y};\boldsymbol{q}\right)$ w.r.t $\boldsymbol{q}$ using $p(\boldsymbol{x})$ from the E-step. For the E-step, the non-negativity of the KL divergence implies that $\mathcal{L}\left(\boldsymbol{y};\boldsymbol{q}\right)$ is maximized when $KL\left(p\left(\boldsymbol{x}\right),p\left(\boldsymbol{x}\big|\boldsymbol{y};\boldsymbol{q}\right)\right)=0$. This is indeed the case, when $p\left(\boldsymbol{x}\right)=p\left(\boldsymbol{x}\big|\boldsymbol{y};\boldsymbol{q}\right)$. The E-step therefore becomes:

$$p^{new}\left(\boldsymbol{x}\right)=p\left(\boldsymbol{x}\big|\boldsymbol{y};\boldsymbol{q}^{j}\right) \tag{2.229}$$

That is, when $p(\boldsymbol{x})$ is equal to the true posterior of $\boldsymbol{x}$ conditioned on $\boldsymbol{q}$ and $\boldsymbol{y}$. Since the entropy is independent of the current values of hyperparameters, the maximization in the M-step is given by:

$$\boldsymbol{q}^{new}=\arg\max_{\boldsymbol{q}}\mathbb{E}_{p^{new}}\left[\ln p\left(\boldsymbol{x},\boldsymbol{y};\boldsymbol{q}\right)\right] \tag{2.230}$$

This process is illustrated in figure 2.11. Here, θ is some hyperparameter to be optimized and the blue curve is the likelihood $p(\boldsymbol{y}|\boldsymbol{q})$. Let θ^{old} (red cross) denote the initial value of the parameter. Then by executing the first E-step, we obtain the lower bound $\mathcal{L}_p(\boldsymbol{y}\big|\boldsymbol{q}^{old})$ corresponding to the red curve. Next, we maximize $\mathcal{L}_p(\boldsymbol{y}\big|\boldsymbol{q})$ w.r.t. $\boldsymbol{q}$ and obtain θ^{new} (green cross). We now perform another E-step and obtain $\mathcal{L}_p(\boldsymbol{y}\big|\boldsymbol{q}^{new})$ corresponding to the green curve and so on. It is seen that the likelihood is increased in each step.

However, the true posterior under the current model is intractable and therefore

Vila et al. uses the approximate posterior provided by GAMP instead of the true posterior. That is, in the E-step for learning the hyperparameters of the prior, the following approximate posterior is used

$$\hat{p}(\boldsymbol{x}\big|\boldsymbol{y},\boldsymbol{q}) = \prod_{i=1}^{n} p(x_i\big|\boldsymbol{y},\boldsymbol{q}) \tag{2.231}$$

When learning the noise variance, the following approximate posterior of z is used in the E-step:

$$\hat{p}(\boldsymbol{z}\big|\boldsymbol{y},\boldsymbol{q}) = \prod_{a=1}^{m} p(z_a\big|\boldsymbol{y},\boldsymbol{q}) \tag{2.232}$$

Both approximate posteriors are readily available after running GAMP.

Furthermore, due to the underlying model, the joint optimization in eq. (2.230) is difficult to perform. Therefore, we will adopt a coordinate-wise maximization scheme for the M-step as in [VS13]. This approach can interpreted as the incremental version of the EM-algorithm [NH98], in which only one hyperparameter is update at a time, while the remaining are held fixed. This means that the EM-updates will be of the form:

$$\lambda^{new} = \arg\max_{\lambda} \mathbb{E}_{p^{new}}\left[\ln p\left(\boldsymbol{x},\boldsymbol{y};\boldsymbol{q}\right)\right] \tag{2.233}$$

where λ is used as example.

Learning Noise Variance

To carry out the M-step for the noise variance, we take the partial derivative of the lower bound $\mathcal{L}_{\hat{p}}(\boldsymbol{y}|\boldsymbol{q})$ w.r.t. σ^2. First, we use the fact that the joint density $p\left(\boldsymbol{x},\boldsymbol{y}\big|\boldsymbol{q}\right)$ can be decomposed into $p\left(\boldsymbol{y};\boldsymbol{x},\boldsymbol{q}\right)p(\boldsymbol{x}|\boldsymbol{q})$:

$$\begin{aligned}
\frac{\partial}{\partial\sigma^2}\mathcal{L}\left(\boldsymbol{y}\big|\boldsymbol{q}\right) &= \frac{\partial}{\partial\sigma^2}\left(\mathbb{E}\left[\ln p\left(\boldsymbol{x},\boldsymbol{y}\big|\boldsymbol{q}\right)\right] + H\left(p\right)\right) \\
&= \frac{\partial}{\partial\sigma^2}\left(\mathbb{E}\left[\ln p\left(\boldsymbol{y}\big|\boldsymbol{x},\boldsymbol{q}\right)p(\boldsymbol{x}\big|\boldsymbol{q})\right] + H\left(p\right)\right) \\
&= \frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\ln p\left(\boldsymbol{y}\big|\boldsymbol{x},\boldsymbol{q}\right)C_1\right] + \frac{\partial}{\partial\sigma^2}C_2
\end{aligned} \tag{2.234}$$

where it is used that both $p(\boldsymbol{x}|\boldsymbol{q})$ and $H(p)$ are independent on σ^2 and can therefore be treated as constants:

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}L(\boldsymbol{y};\boldsymbol{q}) &= C_1\frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\ln p\left(\boldsymbol{y}\middle|\boldsymbol{x},\boldsymbol{q}\right)\right]\\ &= C_1\frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\sum_{a=1}^{m}\ln p\left(y_a\middle|\boldsymbol{x},\boldsymbol{q}\right)\right]\\ &= C_1\sum_{a=1}^{m}\frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\ln p\left(y_a\middle|\boldsymbol{x},\boldsymbol{q}\right)\right]\\ &= C_1\sum_{a=1}^{m}\frac{\partial}{\partial\sigma^2}\int p(z_a|y_a)\ln p\left(y_a\middle|\boldsymbol{x},\boldsymbol{q}\right)\,\mathrm{d}z_a\end{aligned}$$

Using Leibniz' rule for derivatives of integrals, the partial derivative operator can be moved inside the integral:

$$\frac{\partial}{\partial\sigma^2}L(\boldsymbol{y};\boldsymbol{q}) \propto C_1\sum_{a=1}^{m}\int p(z_a|y_a)\frac{\partial}{\partial\sigma^2}\ln p\left(y_a\middle|\boldsymbol{x},\boldsymbol{q}\right)\,\mathrm{d}z_a, \tag{2.235}$$

where it is also used that $p(z_a|y_a)$ is independent of σ^2. Now computing the partial derivative w.r.t. σ^2 is straightforward:

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}\ln p\left(y_a\middle|\boldsymbol{x},\boldsymbol{q}\right) &= \frac{\partial}{\partial\sigma^2}\ln\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_a-z_a)^2}{2\sigma^2}\right)\right]\\ &= -\frac{1}{2\sigma^2}+\frac{(y_a-z_a)^2}{2\left(\sigma^2\right)^2}\\ &= \frac{1}{2}\left(\frac{(y_a-z_a)^2}{\left(\sigma^2\right)^2}-\frac{1}{\sigma^2}\right)\end{aligned}$$

Plugging this result back into eq. (2.235) and rearranging yields:

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}L(\boldsymbol{y};\boldsymbol{q}) &\propto C_1\frac{1}{2}\sum_{a=1}^{m}\int p(z_a|y_a)\left(\frac{(y_a-z_a)^2}{\left(\sigma^2\right)^2}-\frac{1}{\sigma^2}\right)\,\mathrm{d}z_a\\ &= C_1\frac{1}{2}\frac{1}{\left(\sigma^2\right)^2}\sum_{a=1}^{m}\int p(z_a|y_a)\,(y_a-z_a)^2\,\mathrm{d}z_a - C_1\frac{1}{2}\frac{1}{\sigma^2}\sum_{a=1}^{m}\int p(z_a|y_a)\,\mathrm{d}z_a\\ &= C_1\frac{1}{2}\frac{1}{\left(\sigma^2\right)^2}\sum_{a=1}^{m}\int \mathcal{N}\left(z_a\middle|\hat{z}_a,\tau_a^z\right)(y_a-z_a)^2\,\mathrm{d}z_a - C_1\frac{1}{2}\frac{1}{\sigma^2}\sum_{a=1}^{m}\int \mathcal{N}\left(z_a\middle|\hat{z}_a,\tau_a^z\right)\,\mathrm{d}z_a\end{aligned}$$

The integrals are now easily evaluated and the resulting expression is equated to zero:

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}L\left(\boldsymbol{y};\boldsymbol{q}\right) &\propto C_1\frac{1}{2}\frac{1}{\left(\sigma^2\right)^2}\sum_{a=1}^{m}\left(y_a^2+\tau_a^z+\hat{z}_a^2-2y_a\hat{z}_a\right)-C_1\frac{1}{2}\frac{1}{\sigma^2}m\\ &=C_1\frac{1}{2}\frac{1}{\left(\sigma^2\right)^2}\sum_{a=1}^{m}\left[\left(y_a-\hat{z}_a\right)^2+\tau_a^z\right]-C_1\frac{1}{2}\frac{1}{\sigma^2}m=0\end{aligned}$$

and finally, solving for σ^2 gives the update equation:

$$\sigma^2_{\text{new}}=\frac{1}{m}\sum_{a=1}^{m}\left[\left(y_a-\hat{z}_a\right)^2+\tau_a^z\right]$$

Learning the Sparsity Rate

We now repeat the above procedure in order to derive an update equation for learning the sparsity rate λ. However, we have to pay special attention since we have to deal with derivatives of Dirac's delta functions. As before, it is used that the joint density $p\left(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{q}\right)$ can be decomposed to $p\left(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{q}\right)p(\boldsymbol{x}|\boldsymbol{q})$, but now the first factor, i.e. $p\left(\boldsymbol{y};\boldsymbol{x},\boldsymbol{q}\right)$, can be treated as a constant since it is independent of λ. Therefore,

$$\begin{aligned}\frac{\partial}{\partial\lambda}L\left(\boldsymbol{y};\boldsymbol{q}\right) &\propto \sum_{i=1}^{n}\frac{\partial}{\partial\lambda}\mathbb{E}\left[\ln p\left(x_i|\boldsymbol{q}\right)\right]\\ &=\sum_{i=1}^{n}\frac{\partial}{\partial\lambda}\int p(x_i|\boldsymbol{y},\boldsymbol{q})\ln p\left(x_i|\boldsymbol{q}\right)\,\mathrm{d}x_i \qquad (2.236)\end{aligned}$$

In order to apply the Leibniz' rule for exchanging the order of the partial derivative and the integration, it is necessary for both the integrand and its derivative to be continuous w.r.t. λ. But since the prior density $p\left(x_i|\boldsymbol{q}\right)$ is a mixture of a Dirac delta component and a Gaussian component, this does not hold. But in order to justify the application of the rule anyway, the Dirac delta function can be approximated by the continuous function $\mathcal{N}\left(x|0,\epsilon\right)$ for small positive ϵ. This effectively makes the integrand and its partial derivative continuous w.r.t. λ and hence, the Leibniz rule becomes applicable.

$$\frac{\partial}{\partial\lambda}L\left(\boldsymbol{y};\boldsymbol{q}\right)\propto\sum_{i=1}^{n}\int p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{\partial}{\partial\lambda}\ln p\left(x_i|\boldsymbol{q}\right) \qquad (2.237)$$

To compute the partial derivative w.r.t λ, it is necessary to use the same approximation of the Dirac delta function, i.e. $p(x_i|\boldsymbol{q})=(1-\lambda)\mathcal{N}\left(x|0,\epsilon\right)+\lambda\mathcal{N}\left(x|\zeta,\psi\right)$.

This leads to:

$$\begin{aligned}\frac{\partial}{\partial\lambda}\ln p\left(x_i\middle|\boldsymbol{q}\right)\,\mathrm{d}x_i &= \frac{1}{p(x_i|\boldsymbol{q})}\frac{\partial}{\partial\lambda}\left[(1-\lambda)\mathcal{N}\left(x|0,\epsilon\right)+\lambda\mathcal{N}\left(x|\zeta,\psi\right)\right]\\ &= \frac{-\mathcal{N}\left(x|0,\epsilon\right)+\mathcal{N}\left(x|\zeta,\psi\right)}{p(x_i|\boldsymbol{q})} \qquad (2.238)\end{aligned}$$

By taking the limit $\epsilon\to 0$, we have $\mathcal{N}\left(x|0,\epsilon\right)\to\delta(x)$. Therefore:

$$\begin{aligned}\frac{\partial}{\partial\lambda}\ln p\left(x_i\middle|\boldsymbol{q}\right) &= \frac{-\delta(x_i)+\mathcal{N}\left(x_i|\zeta,\psi\right)}{(1-\lambda)\delta(x_i)+\lambda\mathcal{N}\left(x_i|\zeta,\psi\right)}\\ &= \frac{-1+\frac{\mathcal{N}(x_i|\zeta,\psi)}{\delta(x_i)}}{(1-\lambda)+\lambda\frac{\mathcal{N}(x_i|\zeta,\psi)}{\delta(x_i)}} \qquad (2.239)\end{aligned}$$

By analyzing the equations above, it is seen that:

$$\frac{\partial}{\partial\lambda}\ln p\left(x_i\middle|\boldsymbol{q}\right) = \begin{cases}\frac{-1}{1-\lambda} & x_i = 0\\ \frac{1}{\lambda} & x_i \neq 0\end{cases} \qquad (2.240)$$

Vila et al. now employs a neat trick to handle this situation. By introducing the closed ball $\mathbb{B}_\epsilon \equiv [-\epsilon;\epsilon]$ and its complement, $\bar{\mathbb{B}}_\epsilon \equiv \mathbb{R}\setminus\beta_\epsilon$, the domain of integration in eq. (2.237) can be splitted into $\mathbb{B}_\epsilon$ and $\bar{\mathbb{B}}_\epsilon$. Thus,

$$\frac{\partial}{\partial\lambda}L\left(\boldsymbol{y};\boldsymbol{q}\right)\propto\sum_{i=1}^{n}\int_{\mathbb{B}}p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{\partial}{\partial\lambda}\ln p\left(x_i\middle|\boldsymbol{q}\right)+\sum_{i=1}^{n}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{\partial}{\partial\lambda}\ln p\left(x_i\middle|\boldsymbol{q}\right) \qquad (2.241)$$

Then using eq. (2.240), taking the limit $\epsilon\to 0$ and equating the resulting expression to 0 yields:

$$\begin{aligned}\lim_{\epsilon\to 0}\frac{\partial}{\partial\lambda}L\left(\boldsymbol{y};\boldsymbol{q}\right))&\propto -\frac{1}{1-\lambda}\sum_{i=1}^{n}\int_{\mathbb{B}}p(x_i|\boldsymbol{y},\boldsymbol{q})\,\mathrm{d}x_i+\frac{1}{\lambda}\sum_{i=1}^{n}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\,\mathrm{d}x_i = 0\\ &= -\frac{1}{1-\lambda}\sum_{i=1}^{n}(1-\pi_i)+\frac{1}{\lambda}\sum_{i=1}^{n}\pi_i = 0 \qquad (2.242)\end{aligned}$$

Finally, solving for λ yields the intuitive update rule:

$$\lambda^{new} = \frac{1}{n}\sum_{i=1}^{n}\pi_i \qquad (2.243)$$

That is, the updated sparsity rate is simply equal to the mean of the posterior activation probabilities.

Learning the Mean

Next, the update rule for the mean ζ of the Gaussian component of the prior is derived. Starting from eq. (2.236) and applying the same approximation to the Dirac delta function, we arrive at

$$\begin{aligned}\frac{\partial}{\partial\zeta}L(\boldsymbol{y};\boldsymbol{q}) &\propto \sum_{i=1}^{n}\frac{\partial}{\partial\zeta}\int p(x_i|\boldsymbol{y},\boldsymbol{q})\ln p\left(x_i|\boldsymbol{q}\right)\,\mathrm{d}x_i \\ &= \sum_{i=1}^{n}\int p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{\partial}{\partial\zeta}\ln p\left(x_i|\boldsymbol{q}\right)\,\mathrm{d}x_i \end{aligned} \tag{2.244}$$

Using a similar line of arguments as above, we reach:

$$\begin{aligned}\frac{\partial}{\partial\zeta}\ln p\left(x_i|\boldsymbol{q}\right) &= \frac{1}{p(x_i|\boldsymbol{q})}\frac{\partial}{\partial\zeta}\left[(1-\lambda)\delta(x_i)+\lambda\mathcal{N}(x_i|\zeta,\psi)\right] \\ &= \frac{1}{p(x_i|\boldsymbol{q})}\lambda\frac{\partial}{\partial\zeta}\mathcal{N}(x_i|\zeta,\psi) \\ &= \frac{1}{p(x_i|\boldsymbol{q})}\frac{x_i-\zeta}{\psi}\lambda\mathcal{N}(x_i|\zeta,\psi) \\ &= \begin{cases} 0 & x_i = 0 \\ \frac{x_i-\zeta}{\psi} & x \neq 0 \end{cases}\end{aligned} \tag{2.245}$$

Plugging this results into eq. (2.244), and again splitting the domain of integration into the two intervals $\mathbb{B}$ and $\bar{\mathbb{B}}$ yields:

$$\begin{aligned}&\lim_{\epsilon\to 0}\left[\sum_{i=1}^{n}\int_{\mathbb{B}}p(x_i|\boldsymbol{y},\boldsymbol{q})0\,\mathrm{d}x_i+\sum_{i=1}^{n}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{x_i-\zeta}{\psi}\,\mathrm{d}x_i\right] \\ &= \lim_{\epsilon\to 0}\sum_{i=1}^{n}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{x_i-\zeta}{\psi}\,\mathrm{d}x_i \\ &= \frac{1}{\psi}\sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}x_i p(x_i|\boldsymbol{y},\boldsymbol{q})\,\mathrm{d}x_i-\zeta\sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\,\mathrm{d}x_i\end{aligned} \tag{2.246}$$

Equating to zero and solving for ζ gives:

$$\begin{aligned}\zeta &= \frac{\sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}x_i p(x_i|\boldsymbol{y},\boldsymbol{q})\,\mathrm{d}x_i}{\sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\,\mathrm{d}x_i} \\ &= \frac{\sum_{i=1}^{n}\pi_i\gamma_i}{\sum_{i=1}^{n}\pi_i}\end{aligned}$$

Now in the update rule for λ was: $\lambda^{\text{new}} = \frac{1}{n}\sum_{i=1}^{n}\pi_i$, and therefore we can write

$$\zeta^{\text{new}} = \frac{1}{\lambda^{\text{new}} n}\sum_{i=1}^{n}\pi_i\gamma_i, \tag{2.247}$$

That is, ζ is updated using the mean of the active coefficients since the product λn can be interpreted as the effective number of active weights.

Learning the Variance

We now derive an update equation for the last hyperparameter, i.e. the variance of Gaussian component in the prior. Similar to eq. (2.248), the partial derivative of $\mathcal{L}\left(\boldsymbol{y};\boldsymbol{q}\right)$ w.r.t. ψ becomes:

$$\frac{\partial}{\partial\psi}L\left(\boldsymbol{y};\boldsymbol{q}\right) \propto \sum_{i=1}^{n}\int p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{\partial}{\partial\psi}\ln p\left(x_i\middle|\boldsymbol{q}\right)\,\mathrm{d}x_i \tag{2.248}$$

where

$$\begin{aligned}
\frac{\partial}{\partial\psi}\ln p\left(x_i\middle|\boldsymbol{q}\right) &= \frac{1}{p(x_i\middle|\boldsymbol{q})}\frac{\partial}{\partial\psi}\left[(1-\lambda)\delta(x_i)+\lambda\mathcal{N}(x_i|\zeta,\psi)\right]\\
&= \frac{1}{p(x_i\middle|\boldsymbol{q})}\lambda\frac{\partial}{\partial\psi}\mathcal{N}(x_i|\zeta,\psi)\\
&= \frac{\lambda\mathcal{N}(x_i|\zeta,\psi)}{p(x_i\middle|\boldsymbol{q})}\frac{1}{2}\left[\frac{(x-\zeta)^2}{\psi^2}-\frac{1}{\psi}\right]\\
&= \begin{cases} 0 & x_i = 0\\ \frac{1}{2}\left[\frac{(x-\zeta)^2}{\psi^2}-\frac{1}{\psi}\right] & x_i \neq 0\end{cases},
\end{aligned} \tag{2.249}$$

where the partial derivative of $\mathcal{N}\left(x_i\middle|\zeta,\psi\right)$ is obtained using the product rule. Plugging this result into eq. (2.248) and splitting the domain of integration into $\mathbb{B}$ and $\bar{\mathbb{B}}$ yields:

$$\begin{aligned}
\frac{\partial}{\partial\psi}L\left(\boldsymbol{y};\boldsymbol{q}\right) &\propto \sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{1}{2}\left[\frac{(x-\zeta)^2}{\psi^2}-\frac{1}{\psi}\right]\mathrm{d}x_i + \sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\mathbb{B}}p(x_i|\boldsymbol{y},\boldsymbol{q})0\,\mathrm{d}x_i\\
&= \sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\frac{1}{2}\left[\frac{(x-\zeta)^2}{\psi^2}-\frac{1}{\psi}\right]\mathrm{d}x_i\\
&= \frac{1}{2}\frac{1}{\psi^2}\sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\left(x-\zeta\right)^2\mathrm{d}x_i - \frac{1}{2}\frac{1}{\psi}\sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\,\mathrm{d}x_i\\
&= \frac{1}{2}\frac{1}{\psi^2}\sum_{i=1}^{n}\lim_{\epsilon\to 0}\int_{\bar{\mathbb{B}}}p(x_i|\boldsymbol{y},\boldsymbol{q})\left(x-\zeta\right)^2\mathrm{d}x_i - \frac{1}{2}\frac{1}{\psi}\sum_{i=1}^{n}\pi_i
\end{aligned}$$

Expanding the parenthesis: $(x-\zeta)^2 = x_i^2 + \zeta^2 - 2x_i\zeta$ and carrying out the integration for each term gives:

$$\frac{\partial}{\partial\psi} L\left(\boldsymbol{y};\boldsymbol{q}\right) \propto= \frac{1}{1}\frac{1}{\psi^2}\sum_{i=1}^{n}\left[\pi_i\left(\nu_i + \gamma_i + \zeta^2 - 2\zeta\gamma_i\right)\right] - \frac{1}{2}\frac{1}{\psi}\sum_{i=1}^{n}\pi_i \tag{2.250}$$

and finally, equating the resulting expression to zero and solving for ψ:

$$\begin{aligned}\psi^{\text{new}} &= \frac{1}{\sum_{i=1}^{n}\pi_i}\sum_{i=1}^{n}\left[\pi_i\left(\nu_i + \gamma_i + \zeta^2 - 2\zeta\gamma_i\right)\right] \\ &= \frac{1}{\lambda^{\text{new}} n}\sum_{i=1}^{n}\left[\pi_i\left(\zeta-\gamma_i\right)^2 + \nu_i\right],\end{aligned} \tag{2.251}$$

which is the last update equation for the EM scheme.

Initialization of the Hyperparameters

Since the EM-algorithm is only guaranteed to converge to a stationary point, the initialization of the hyperparameter often have a crucial effect on the performance. In [VS13], Vila et al. provide an initialization scheme, which are claimed to provide good empirical results. They suggest that the initial sparsity rate λ^0 is set equal to the theoretical phase transition curve for ℓ_1 minimization (see Literature Review in sec. 1.2). That is,

$$\lambda^0 = \delta \cdot \rho_{\text{SE}}(\delta) \tag{2.252}$$

We have to multiply by δ to convert from ρ-sparsity to λ-sparsity. Furthermore, if an initial estimate of the signal-to-noise ratio is available SNR^0, the variances of the noise and the variance of the "slap"-component should be initialized as:

$$\sigma_0^2 = \frac{||\boldsymbol{y}||_2^2}{(\text{SNR}^0+1)m} \qquad \psi^0 = \frac{||\boldsymbol{y}||_2^2 - m\sigma_0^2}{||\boldsymbol{A}||_F^2\,\lambda^0} \tag{2.253}$$

If an estimate of the SNR is not available, they just put SNR$^0 = 100$ (not in dB). Finally, the mean value of the "slap"-component is simply initialized as $\zeta^0 = 0$. The entire EMBGAMP algorithm is summarized in Algorithm 5. Notice, the complete EMBGAMP algorithm has two layers of nested iterations. The inner layer is the GAMP-iterations and the outer layer is the EM-iterations.

Example: Toy Problem

The EMBGAMP procedure is now illustrated using an example similar to the one used in the AMP example in section 2.2. Consider a noisy problem with

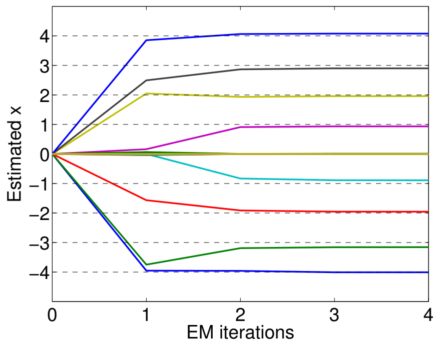
Algorithm 5 EMBGAMP algorithm (EMBGAMP)

```
• Initialize the hyperparameters:
        $\sigma_0^2 = \frac{||\boldsymbol{y}||_2^2}{(\mathrm{SNR}^0 + 1)m}$
        $\lambda^0 = \delta \cdot \rho_{\mathrm{SE}}(\delta)$
        $\psi^0 = \frac{||\boldsymbol{y}||_2^2 - m\sigma_0^2}{||\boldsymbol{A}||_F^2 \, \lambda^0}$
        $\zeta^0 = 0$
• repeat until stopping criteria:
        Run BG-AMP and obtain the quantities: $\hat{\boldsymbol{x}}, \boldsymbol{\pi}, \hat{\boldsymbol{z}}, \boldsymbol{\tau}^z, \boldsymbol{\gamma}, \boldsymbol{\nu}$
        Test for convergence or for maximum number of iterations
        If not converged, update hyperparameters:
            $\lambda^{new} = \frac{1}{n} \sum_{i=1}^{n} \pi_i$
            $\zeta^{\mathrm{new}} = \frac{1}{\lambda^{\mathrm{new}} n} \sum_{i=1}^{n} \pi_i \gamma_i$
            $\psi^{\mathrm{new}} = \frac{1}{\lambda^{\mathrm{new}} n} \sum_{i=1}^{n} \left[\pi_i \left(\zeta - \gamma_i\right)^2 + \nu_i\right]$
            $\sigma_{\mathrm{new}}^2 = \frac{1}{m} \sum_{a=1}^{m} \left[\left(y_a - \hat{z}_a\right)^2 + \tau_a^z\right]$
```

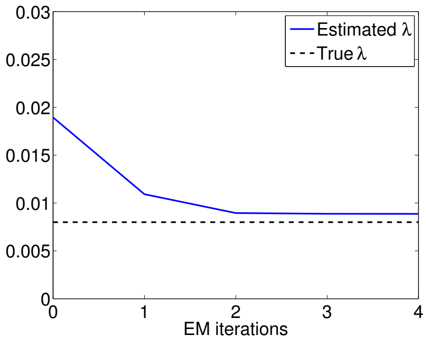
$n = 1000$, $m = 100$, $k = 8$ and SNR$= 20dB$. Let the true solution $\boldsymbol{x}_0$ be defined as:

$$\boldsymbol{x}_0 = \begin{bmatrix} -4 & -3 & -2 & -1 & 1 & 2 & 3 & 4 & 0 & 0.. \end{bmatrix}^T \in \mathbb{R}^n \tag{2.254}$$

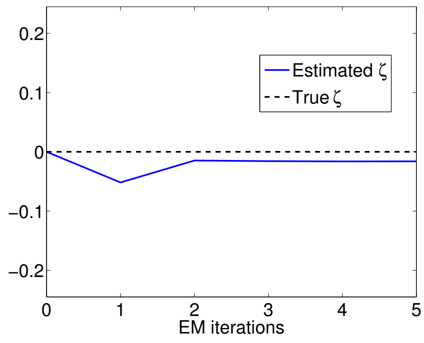
The measurements are generated using $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e}$, where A_{ai} are I.I.D. as $A_{ai} \sim \mathcal{N}(0, 1/m)$ and e_a is I.I.D. as $e_a \sim \mathcal{N}\left(0, \sigma^2\right)$, where σ^2 is scaled to yield SNR$= 20dB$. Figure 2.12(a) shows the estimated coefficients of $\hat{\boldsymbol{x}}$ as a function of the EM iterations, when the EMBGAMP algorithm is applied to the problem. The dashed lines indicates the true coefficients. The estimated coefficients are initialized at 0. It is seen that they converge in approximately 4 EM iterations to values close to the true values. Figure 2.12(b) shows the evolution of the hyperparameter λ, which is also seen to converge to a value close to the true value. The figures 2.12(c)-(e) show similar plots for the remaining hyperparameter, where it is seen that all the estimated hyperparameters converge reasonable values. Keep in mind that due to $k = 8$, the algorithm has only 8 samples to estimate the prior statistics.



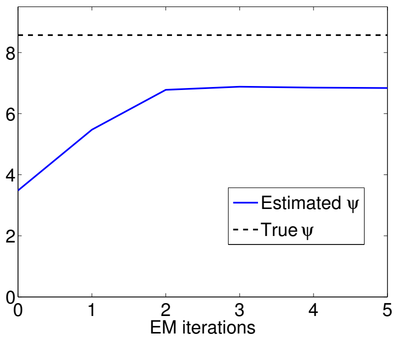
(a) Estimated coefficients



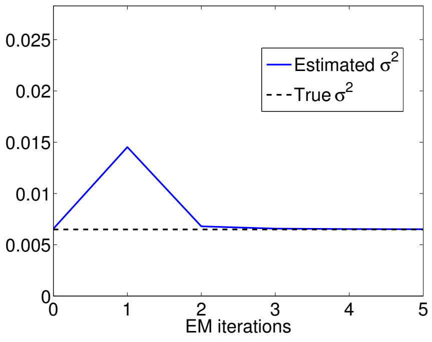
(b) Estimated sparsity



(c) Estimated signal mean



(d) Estimated signal variance



(e) Estimated noise variance

Figure 2.12: Illustration of the EMBGAMP using a toy problem with dimensions: $n = 1000$, $\delta = 0.1$, $k = 8$, SNR$= 20dB$. It is shown how the estimated solution and hyperparameter evolve as a function of EM iterations. The dashed lines indicates true values

Chapter 3

Theory: The Multiple Measurement Vector Problem

The aim of this chapter is to extend the BG-AMP algorithm and the accompanying Expectation-Maximization scheme to the MMV formulation. In particular, two different algorithms will be derived: AMP-MMV and AMP-DCS. The AMP-MMV algorithm (section 3.1) extends the BG-AMP algorithm to the MMV problem using the common sparsity assumption, while the AMP-DCS algorithm (section 3.2) is designed to handle problems, where the support is assumed to change slowly over time.

3.1 AMP-MMV: Assuming Static Support

The BG-AMP algorithm was derived by combining the GAMP algorithm with a Bernoulli-Gaussian prior distribution. In this section, this algorithm is extended to the multiple measurement vector problem (MMV) using the common sparsity assumption (see literature review in chapter 1.2) based on the work of Ziniel et al. in [ZS13b]. The extended algorithm is referred to as *AMP-MMV*.

Consider the consecutive sequence of linear inverse problems at time $t = 1, .., T$:

$$\boldsymbol{y}^t = \boldsymbol{A}\boldsymbol{x}^t + \boldsymbol{e}^t \qquad \text{for} \qquad t = 1, .., T \tag{3.1}$$

Assuming that the forward matrix $\boldsymbol{A}$ does not depend on the time index t, the measurement vectors $\{\boldsymbol{y}^t\}_{t=1}^T$ can be concatenated into a measurement matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times T}$, the true solution vectors $\{\boldsymbol{x}^t\}_{t=1}^T$ can be concatenated into a solution matrix $\boldsymbol{X} \in \mathbb{R}^{n\times T}$ and errors vector $\{\boldsymbol{e}^t\}_{t=1}^T$ are stacked into an error matrix $\boldsymbol{E} \in \mathbb{R}^{m\times T}$. Using this notation the MMV problem can be reformulated as:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{E} \tag{3.2}$$

The common sparsity assumption implies that the support of the solutions $\{\boldsymbol{x}^t\}_{t=1}^T$ is constant in time. Let s_i^t be an indicator variable for the support of x_i^t and let θ_i^t be the corresponding coefficient or amplitude. The common sparsity assumption then implies that:

$$s_i^t = s_i \qquad \forall t \in [T] \tag{3.3}$$

Therefore, the i'th entry of the signal $\boldsymbol{x}^t$ at time t can be decomposed as:

$$x_i^t = s_i\theta_i^t \qquad \Longleftrightarrow \qquad p(x_i^t | s_i, \theta_i^t) = \begin{cases} \delta\left(x_i^t\right) & \text{if } s_i = 0 \\ \delta\left(x_i^t - \theta_i^t\right) & \text{if } s_i = 1 \end{cases}, \tag{3.4}$$

where it is noted that the support variables s_i do not depend on the time index t.

It is also reasonable to assume that the coefficients of the estimated solution, θ_i^t, contain some degree of correlation across time. Assuming independence of correlated measurement vectors have been shown to degrade performance [ZR13]. Hence, it is of interest to model this temporal correlation structure as well. However, the degree of correlation is usually not known in advance and therefore have to be modelled as well. Ziniel et al. implements temporal correlation using a stationary first-order Gauss-Markov process of the form

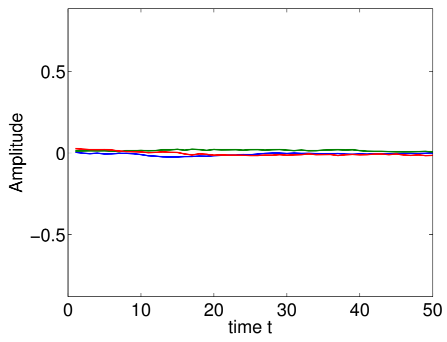
$$\theta_i^t = (1-\alpha)\left(\theta_i^{t-1} - \zeta\right) + \alpha w_i^t + \zeta, \tag{3.5}$$

where $\zeta \in \mathbb{R}$ is the mean of the process, $w_i^t \sim \mathcal{N}(0, \rho)$ is the driving process and $\alpha \in [0, 1]$ governs the degree of temporal correlation of the process. It is seen that when α takes the extreme value 1, the process is simply uncorrelated noise and we have $\theta_i^t \sim \mathcal{N}(\zeta, \rho)$ and in the other extreme, i.e. $\alpha = 0$, the process is constant. The relation in eq. (3.5) implies that the conditional density of θ^t conditioned on θ^{t-1} is given by

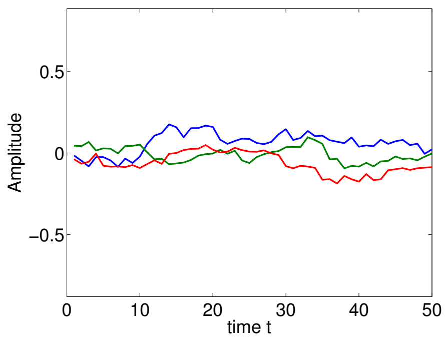
$$p(\theta^t | \theta^{t-1}) = \mathcal{N}\left(\theta^t \middle| (1-\alpha)\left(\theta^{t-1} - \zeta\right) + \zeta,\, \alpha^2\rho\right) \tag{3.6}$$

This shows that the process satisfies the Markov property [PK11]. The conditional densities are Gaussian for all values of $t \in [T]$ and this implies that the process indeed is a Gaussian process [Bis06]. Hence, the name of the process.

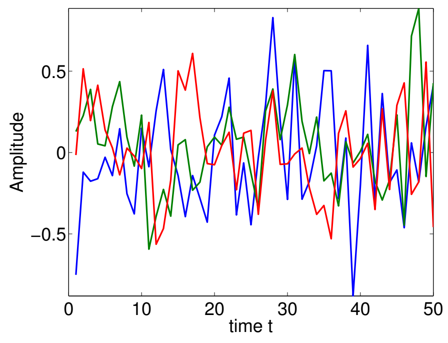
Due to the construction of this particular signal model, there exists a non-zero coefficient θ_i^t for x_i^t even when $s_i = 0$. Consequently, the coefficient θ_i^t should be interpreted as being the coefficient of x_i^t given $s_i = 1$.



(a) $\alpha = 0.01$



(b) $\alpha = 0.10$



(c) $\alpha = 0.90$

Figure 3.1: Realizations of the Gauss-Markov process, which models the temporal correlation of the coefficients. Each plot shows 3 instances generated from a Gauss-Markov process with parameters $\zeta = 0$ and $\rho = 0.1$. The values of α for the three different plots are $0.01, 0.10, 0.90$, respectively.

Figure 3.1 shows realizations of the Gauss-Markov process for the different values of $\alpha \in \{0.01, 0.10, 0.90\}$. Each of the three subplots shows three instances of the process for the parameters $\zeta = 0$ and $\rho = 0.1$. In the left-most plot, $\alpha = 0.01$ and it is seen that the process is almost constant. The realizations in the center plot is for $\alpha = 0.10$ and a high degree of temporal correlation is observed. On the contrary, the right-most plot shows 3 realizations for $\alpha = 0.9$ and here it is seen that the observed processes almost resembles independent noise.

Similar to the BG-AMP model, let λ denote the marginal probability of a variable being active, i.e. $\lambda \equiv p(s_i = 1)$. This leads to a marginal prior distribution on x_i^t given by:

$$p(x_i^t) = (1-\lambda)\delta(x_i^t) + \lambda\mathcal{N}\left(x_i^t \middle| \zeta, \sigma_c^2\right), \tag{3.7}$$

where $\sigma_c^2 = \frac{\alpha\rho}{2-\alpha}$ is the steady-state variance of the coefficients, θ_i^t. The noise is still assumed to be independent and Gaussian distribution with noise parameter σ^2.

Let $\boldsymbol{s}$ be a binary vector indicating the locations of support, i.e. $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & \dots & s_n \end{bmatrix}$ and let $\bar{\boldsymbol{\theta}} \in \mathbb{R}^{n\times T}$ be a matrix containing the signal coefficients, i.e. $\bar{\boldsymbol{\theta}}_{it} = \theta_i^t$. Then the complete joint density is given by:

$$\begin{aligned} p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{s}, \bar{\boldsymbol{\theta}}) &= \prod_{t=1}^{T} p(\boldsymbol{y}^t|\boldsymbol{x}^t)p(\boldsymbol{x}^t|\boldsymbol{s}, \boldsymbol{\theta}^t)p(\boldsymbol{s})p(\boldsymbol{\theta}^t\big|\boldsymbol{\theta}^{t-1}) \\ &= \prod_{t=1}^{T}\left(\prod_{a=1}^{m} p(y_a^t|\boldsymbol{x}^t)\prod_{i=1}^{n} p(x_i^t|s_i, \theta_i^t)p(\theta_i^t\big|\theta_i^{t-1})\right)\prod_{i=1}^{n} p(s_i) \end{aligned}$$

where $p(\theta_i^1|\theta_i^0) = \mathcal{N}\left(\theta_i^t \middle| \zeta, \sigma_c^2\right)$ for all i.

Figure 3.2 shows the resulting factor graph for $T = 3$. But note that for the sake of visual clarity, the dependencies between two consecutive time steps are only included for variables s_1^t and θ_1^t for $t = 1, 2, 3$, but the remaining variables do, of course, have similar connections. The factor graph consists of multiple subgraphs corresponding to each *time frame* t (red boxes) and a set of nodes implementing the temporal structure, i.e. the constant support and temporal correlation of the coefficients.

Notice, the subgraphs inside the blue boxes correspond exactly to the factor graph for the BG-AMP model from the previous section and the BG-AMP algorithm expects a prior of the form given in eq. (3.7). The idea is therefore to use conventional sum-product message passing to propagate messages across time frames to update the "local priors", i.e. the factor nodes $p(x_i^t|\theta_i^t, s_i)$ inside the blue boxes, and then use BG-AMP algorithm handle the message passing within the blue boxes. This idea of using AMP as a component in a larger algorithm has earlier been used by Philip Schniter in [Sch10].

Because the factor graph in figure 3.2 contains multiple loops, there are many ways to *schedule* the messages. Ziniel et al. discusses two different scheduling

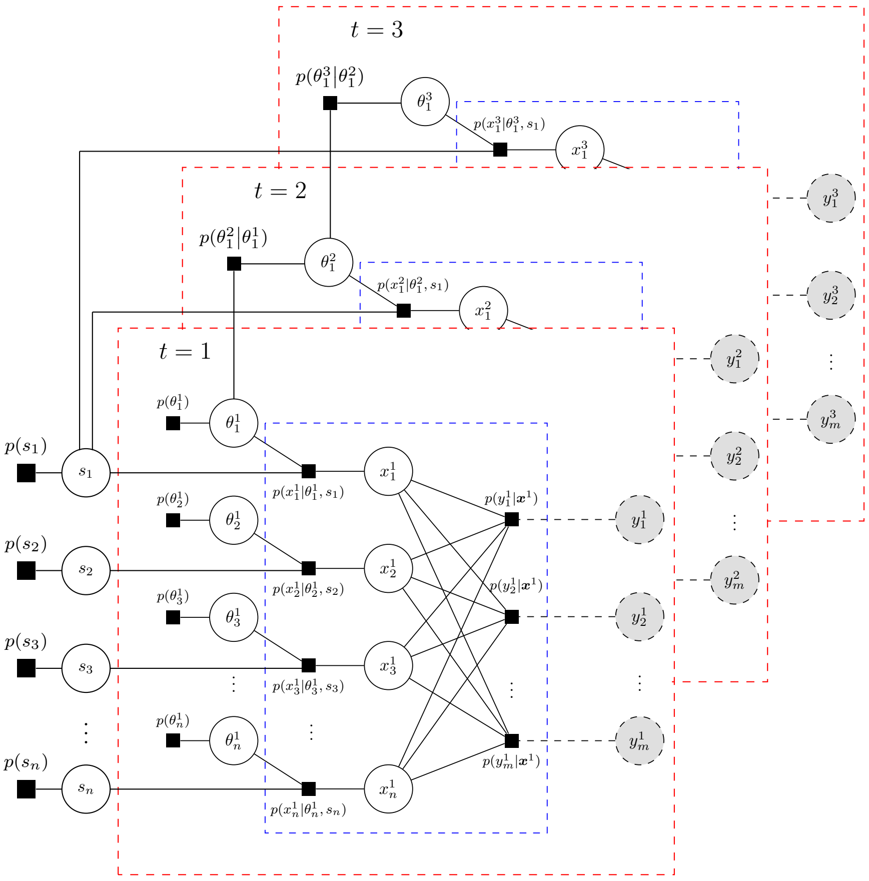


Figure 3.2: Factor graph for MMV model. For the visual sake of clarity, the dependencies between two consecutive time frames are only included for the s_i and θ_i^t variables for x_1^t for $t = 1, 2, 3$. The messages propagated within the blue boxes are handled using the GAMP framework, while the remaining messages are handled explicitly.

schemes, which are claimed to provide good empirical convergence properties [ZS13a]. That is, the *serial* and the *parallel* scheduling schemes. For both schemes, the message passing process is divided into the four phases, which are common for both scheduling schemes. The two schemes then only differ in the order of execution of those four phases. The phases are given the names: *(into)*, *(within)*, *(out)* and *(across)*.

The (into) phase is responsible for setting up the local prior. That is, the current belief about the support and the coefficients is updated by propagating the messages from variable nodes s_i and $\theta_i^t\ \forall i$ into frame t. The (within) phase simply corresponds to running the BG-AMP algorithm using the updated local prior, i.e. in this phase the estimate of $\boldsymbol{x}^t$ is refined using $(\boldsymbol{s}, \boldsymbol{\theta}^t, \boldsymbol{y}^t)$. In the (out) phase, the current estimate of $\boldsymbol{x}^t$ is used to update the belief about the support and the coefficients by propagating messages from node x_i^t to the node s_i and θ_i^t for all i. In the last phase, (across), the current belief is propagated between consecutive frames. That is, messages are sent from θ_i^t to either θ_i^{t+1} or θ_i^{t-1}.

In the serial scheduling scheme, the time frames (the big dashed boxes in figure 3.2) are processed in a sequential manner. First, the messages are propagated into the first frame, i.e. $t = 1$ by executing the *(into)*-phase, then the measurements for this time frame are processed in the *(within)* phase, followed by the execution of the *(out)*-phase and finally, the messages propagated to the consecutive frame, i.e. frame $t = 2$ in the *(across)* phase. This sequence is then repeated until the final frame $t = T$ is reached. This entire sequence is denoted a *forward pass*. If the data are available offline, the messages can also be propagated back again. That is, we propagate messages from time frame $t = T$ to time frame $t = T - 1$ in the same manner until we reach the first frame again, i.e. $t = 1$. When we reach frame $t = 1$ again, we have completed a complete *forward/backward pass*. The process can be interpreted as a smoothing process and it can be repeated until parameters converge or until a maximum number of iterations is reached. The serial scheme also provide the potential for *causal filtering* of MMV signals by only performing the forward pass. This is very usable in online applications like real-time EEG.

In the parallel scheme, the *(into)*-phase is executed for all frames $t = 1, .., T$ simultaneously, then the*(within)*-phases are executed for all the frames and followed by the *(out)* phases for all the frames. Then the *(across)*-phases are executed starting from frame $t = 1$, and the propagating messages all the way to frame $t = T$. After reaching frame $t = T$, the *(across)*-phases are now executed in the opposite order. That is, starting from frame $t = T$ and ending in frame $t = 1$. Once both the forward and backward pass have been completed, a single iteration of parallel AMP-MMV has been completed. In this thesis, we will limit ourselves to consider the serial scheme, since this also enables causal filtering of the data.

The following subsections describes how the messages for the four phases are derived using the sum-product algorithm. We will focus on a single variable x_i^t at time $1 < t < T$ in the forward direction, i.e. from time t to time $t+1$.

The (into) Phase

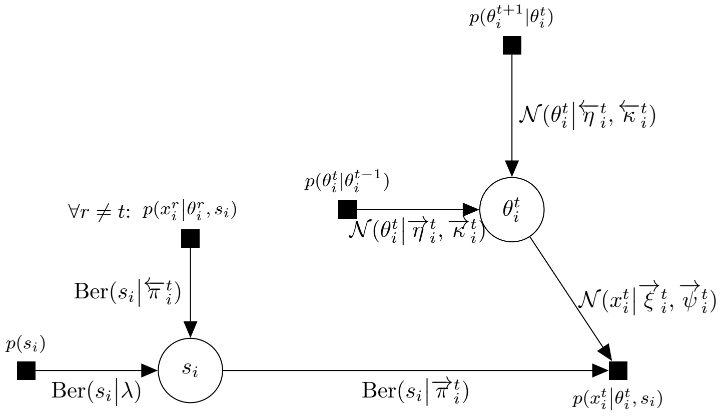

$p(\theta_i^{t+1}|\theta_i^t)$
$\mathcal{N}(\theta_i^t|\overleftarrow{\eta}_i^t, \overleftarrow{\kappa}_i^t)$
$p(\theta_i^t|\theta_i^{t-1})$
$\mathcal{N}(\theta_i^t|\overrightarrow{\eta}_i^t, \overrightarrow{\kappa}_i^t)$
θ_i^t
$\forall r \neq t$: $p(x_i^r|\theta_i^r, s_i)$
$\text{Ber}(s_i|\overleftarrow{\pi}_i^t)$
$\mathcal{N}(x_i^t|\overrightarrow{\xi}_i^t, \overrightarrow{\psi}_i^t)$
$p(s_i)$
s_i
$\text{Ber}(s_i|\lambda)$
$\text{Ber}(s_i|\overrightarrow{\pi}_i^t)$
$p(x_i^t|\theta_i^t, s_i)$

Figure 3.3: The factor sub graph for the (into) phase for the variable x_i^t at time t.

Figure 3.3 shows the corresponding subgraph for the (into)-phase as well as the functional form and parametrization of the message associated to each edge of the subgraph. Since messages are propagated along most edges in both directions, arrows indicate the current direction of a given message for the specific phase.

Consider the message from variable node s_i to factor node $p(x_i^t|\theta_i^t, s_i)$. As we will see soon, this message corresponds to a Bernoulli density and can be parametrized by $\overrightarrow{\pi}$, which denotes the probability of $s_i = 1$ under this density.

Applying the standard sum-product rules gives rise to the following message:

$$\begin{aligned}\mu_{s_i \to p(x_i^t|\theta_i^t, s_i)}(s_i) &= \mu_{p(s_i)\to s_i}(s_i) \prod_{r\neq t} \mu_{p(x_i^r|\theta_i^r, s_i)\to s_i}(s_i) \\ &= \left[(1-\lambda)(1-s_i) + \lambda s_i\right] \prod_{r\neq} \left[(1-\overleftarrow{\pi}_i^r)(1-s_i) + \overleftarrow{\pi}_i^r s_i\right]\end{aligned}$$

Note that all "cross terms", i.e. terms including both s_i and $(1 - s_i)$, will be zero since $s_i \in \{0, 1\}$. Therefore,

$$\begin{aligned}\mu_{s_i \to p(x_i^t | \theta_i^t, s_i)}(s_i) &= \left[(1-\lambda)\prod_{r\neq}(1-\overleftarrow{\pi}_i^r)\right](1-s_i) + \left[\lambda\prod_{r\neq t}\overleftarrow{\pi}_i^r\right]s_i \\ &\propto (1-\overrightarrow{\pi}_i^t)(1-s_i) + \overrightarrow{\pi}_i^t s_i \qquad (3.8)\end{aligned}$$

Thus, this message is proportional to a Bernoulli density parametrized by $\overrightarrow{\pi}_i^t$ given by:

$$\overrightarrow{\pi}_i^t = \frac{\lambda\prod_{r\neq t}\overleftarrow{\pi}_i^r}{(1-\lambda)\prod_{r\neq t}(1-\overleftarrow{\pi}_i^r) + \lambda\prod_{r\neq t}\overleftarrow{\pi}_i^r} \qquad (3.9)$$

Informally, this probability is updated by combining the belief from prior $p(s_i)$ with the current beliefs from the other time frames. When processing the first frame, i.e. $t = 1$, we have no knowledge about the subsequent frames and therefore the parameters are initialized by $\overleftarrow{\pi}_i^t = 0.5$ for all $i \in [n]$ and $t \in [T]$. This implies that $\overrightarrow{\pi}_i^1 = \lambda$ for all i.

Next, we consider the message from variable node θ_i^t to factor node $p(x_i^t | \theta_i^t, s_i)$. This message contains the combined belief about θ_i^t from the previous and the subsequent time frame:

$$\begin{aligned}\mu_{\theta_i^t \to p(x_i^t | \theta_i^t, s_i)}(\theta_i^t) &= \mu_{p(\theta_i^{t-1}) \to \theta_i^t}(\theta_i^t)\mu_{p(\theta_i^{t+1}) \to \theta_i^t}(\theta_i^t) \\ &= \mathcal{N}\left(\theta_i^t | \overrightarrow{\eta}_i^t, \overrightarrow{\kappa}_i^t\right)\mathcal{N}\left(\theta_i^t | \overleftarrow{\eta}_i^t, \overleftarrow{\kappa}_i^t\right) \\ &\propto \mathcal{N}\left(\theta_i^t | \overrightarrow{\xi}_i^t, \overrightarrow{\psi}_i^t\right), \qquad (3.10)\end{aligned}$$

where the parameters, $\overrightarrow{\xi}_i^t$ and $\overrightarrow{\psi}_i^t$, are easily obtained using the Gaussian multiplication rule (see Appendix A.1):

$$\overrightarrow{\xi}_i^t = \left(\frac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \frac{\overleftarrow{\eta}_i^t}{\overleftarrow{\kappa}_i^t}\right) \cdot \overrightarrow{\psi}_i^t \qquad \overrightarrow{\psi}_i^t = \frac{1}{\frac{1}{\overrightarrow{\kappa}_i^t} + \frac{1}{\overleftarrow{\kappa}_i^t}} \qquad (3.11)$$

Analogous to support probabilities, when processing the first frame, i.e. $t = 1$, we have no knowledge of subsequent frames and therefore, we use the following initialization to avoid any influence from the parameters for the subsequent frames:

$$\overleftarrow{\eta}_i^t = 0 \qquad \overleftarrow{\kappa}_i^t = \infty \qquad \forall i \in [n], \forall t \in [T] \qquad (3.12)$$

and for the first frame, i.e. $t = 1$, we simply use the hyperparameters of the marginal prior:

$$\overrightarrow{\eta}_1^t = \zeta \qquad \overrightarrow{\kappa}_1^t = \frac{\alpha\rho}{2-\alpha},$$

This completes the (into) phase. That is, executing the (into) phase simply corresponds updating the parameters using eq. (3.9), eq. (3.10), and eq. (3.11). We will now move on to the (within) phase.

The (within) Phase

This phase simple correspond to running the BG-AMP algorithm (Algorithm 4) using the hyperparameters obtained from the (into)-phase $(\overrightarrow{\pi}_i^t, \overrightarrow{\xi}_i^t, \overrightarrow{\psi}_i^t), \forall i \in [n]$ and the noise variance σ^2. However, the BG-AMP algorithm was based on GAMP1. Here we will utilize the GAMP2 algorithm rather than GAMP1 due to the significantly increased complexity. Thus, from the BG-AMP algorithm, we obtain $\hat{r}_i^t$ and the corresponding scalar variances $\tau^r(t)$.

The (out) Phase

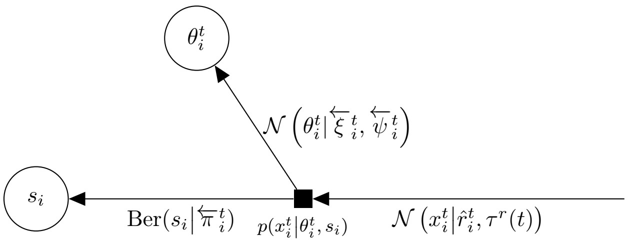

θ_i^t
$\mathcal{N}\left(\theta_i^t | \overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right)$
s_i
$\text{Ber}(s_i | \overleftarrow{\pi}_i^t)$
$p(x_i^t|\theta_i^t, s_i)$
$\mathcal{N}\left(x_i^t | \hat{r}_i^t, \tau^r(t)\right)$

Figure 3.4: The factor sub graph for the (out) phase for the variable x_i^t at time t.

After completing the (within) phase, the updated beliefs must be propagated out of the current time frame. The relevant subgraph is shown in figure 3.4. For the messages from variable node x_i^t to factor node $p(x_i^t|\theta_i^t, s_i)$, the results from the BG-AMP algorithm is used. That is,

$$\mu_{x_i^t \to p(x_i^t|\theta_i^t, s_i)}(x_i^t) = \mathcal{N}\left(x_i^t \middle| \hat{r}_i^t, \tau^r(y)\right) \tag{3.13}$$

The message from factor node $p(x_i^t|\theta_i^t, s_i)$ to variable node s_i can now be computed:

$$\begin{aligned}
\mu_{p(x_i^t|\theta_i^t,s_i)\to s_i}(s_i) &= \int p(x_i^t|\theta_i^t,s_i)\mu_{\theta_i^t\to p(x_i^t|\theta_i^t,s_i)}(\theta_i^t)\mu_{x_i^t\to p(x_i^t|\theta_i^t,s_i)}(x_i^t)\,\mathrm{d}x_i^t\,\mathrm{d}\theta_i^t \\
&= \int \delta\left(x_i^t - s_i\theta_i^t\right)\mathcal{N}\left(\theta_i^t|\overrightarrow{\xi},\overrightarrow{\psi}\right)\mathcal{N}\left(x_i^t|\hat{r}_i^t,\tau^r(t)\right)\,\mathrm{d}x_i^t\,\mathrm{d}\theta_i^t \\
&= \int \mathcal{N}\left(\theta_i^t|\overrightarrow{\xi},\overrightarrow{\psi}\right)\mathcal{N}\left(s_i\theta_i^t|\hat{r}_i^t,\tau^r(t)\right)\,\mathrm{d}\theta_i^t,
\end{aligned}$$

where the sift property of Dirac's delta function is used in the last step. Before the integration over θ_i^t is carried out, the expression is divided into the two cases $s_i = 0$ and $s_i = 1$, respectively, and then the Gaussian multiplication rule is applied (see Appendix A.1). This leads to

$$\begin{aligned}
\mu_{p(x_i^t|\theta_i^t,s_i)\to s_i}(s_i) &= \mathcal{N}\left(0|\hat{r}_i^t,\tau^r(t)\right)(1-s_i) + \mathcal{N}\left(0|\hat{r}_i^t - \overrightarrow{\xi},\tau^r(t)+\overrightarrow{\psi}_i^t\right)s_i \\
&\propto \left(1-\overleftarrow{\pi}_i^t\right)(1-s_i) + \overleftarrow{\pi}_i^t s_i,
\end{aligned} \tag{3.14}$$

where $\overleftarrow{\pi}_i^t$ is given by[1]

$$\overleftarrow{\pi}_i^t = \frac{\mathcal{N}\left(0|\hat{r}_i^t - \overrightarrow{\xi}_i^t,\tau^r(t)+\overrightarrow{\psi}_i^t\right)}{\mathcal{N}\left(0|\hat{r}_i^t,\tau^r(t)\right) + \mathcal{N}\left(0|\hat{r}_i^t - \overrightarrow{\xi}_i^t,\tau^r(t)+\overrightarrow{\psi}_i^t\right)} \tag{3.15}$$

where we note the similarity to the posterior activation probability derived in eq. (2.216).

For the message from the factor node $p(x_i^t|\theta_i^t,s_i)$ to variable θ_i^t, the sum-product rules yields:

$$\begin{aligned}
\mu_{p(x_i^t|\theta_i^t,s_i)\to\theta_i^t}(\theta_i^t) &= \sum_{s_i}\int \delta\left(x_i^t - s_i\theta_i^t\right)\mathcal{N}\left(x_i^t|\hat{r}_i^t,\tau^r(t)\right)\left[\left(1-\overrightarrow{\pi}_i^t\right)(1-s_i) + \overrightarrow{\pi}_i^t s_i\right]\mathrm{d}x_i^t \\
&= \left(1-\overrightarrow{\pi}_i^t\right)\mathcal{N}\left(0|\hat{r}_i^t,\tau^r(t)\right) + \overrightarrow{\pi}_i^t\mathcal{N}\left(\theta_i^t|\hat{r}_i^t,\tau^r(t)\right)
\end{aligned} \tag{3.16}$$

We now notice that the message is improper, i.e. it is not normalizable due to the first term being constant w.r.t. θ_i^t. As Ziniel et al. point out in [ZS13b], this is caused by the fact that for $s_i = 0$, x_i^t does not provide any information about the coefficient θ_i^t.

To avoid this issue Ziniel et al. consider the signal model $s_i \in \{0,1\}$ as the limiting case of the model $s_i \in \{\epsilon,1\}$ for $\epsilon \to 0$. Then according to [ZS13b], for

[1] In both papers [ZS13b, ZS13a] in the definition of γ, there seem to be missing a factor, $\frac{1}{2}$, inside the exponential.

any fixed positive ϵ, the normalized message become:

$$\bar{\mu}_{p(x_i^t|\theta_i^t,s_i)\to\theta_i^t}(\theta_i^t) = \left(1-\Omega\left(\overrightarrow{\pi}_i^t\right)\right)\mathcal{N}\left(\theta_i^t|\frac{1}{\epsilon}\hat{r}_i^t, \frac{1}{\epsilon^2}\tau^r(t)\right) + \Omega\left(\overrightarrow{\pi}_i^t\right)\mathcal{N}\left(\theta_i^t|\hat{r}_i^t, \tau^r(t)\right), \tag{3.17}$$

where the function $\Omega(\pi)$ is defined as:

$$\Omega(\pi) \equiv \frac{\epsilon^2\pi}{(1-\pi)+\epsilon^2\pi} \tag{3.18}$$

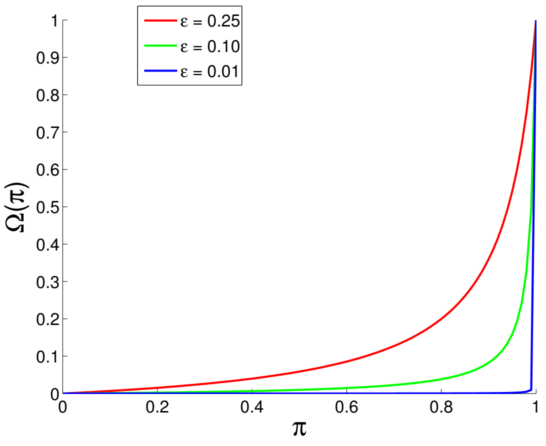


Figure 3.5: Plot of $\Omega_\epsilon(\pi)$ for 3 different values of $\epsilon \in \{0.25, 0.10, 0.01\}$. It is seen that as ϵ approaches 0, the function $\Omega_\epsilon(\pi)$ becomes an indicator function, $\mathbb{I}[\pi = 1]$.

Figure 3.5 shows a plot of $\Omega(\pi)$ for 3 different values of $\epsilon \in \{0.25, 0.10, 0.01\}$. It is seen that the function behaves intuitively pleasing, since $\Omega(\pi)$ approach an indicator function with argument $\pi = 1$ when ϵ approach 0. That is, $\Omega(\pi) \to \mathbb{I}[\pi = 1]$ for $\epsilon \to 0$. Ziniel et al. advocates using a small value for ϵ, like $\epsilon = 10^{-7}$ [ZS13b].

This message is now normalizable and recognized as a Gaussian mixture with two components. However, when a Gaussian mixture is propagated along a given edge, it leads to an exponential growth in the number of mixture components for subsequent edges, which is of course intractable.

To solve this issue, Ziniel et al. approximates this message using a single Gaussian component[2]. This is justified by the fact that for $\epsilon << 1$, the function $\Omega(\pi)$

[2]This is sometimes referred to as a Gaussian sum approximation. [AS72]

behaves like an indicator function and therefore one of the Gaussian components are likely to have negligible mass. The approximation is implemented by using a second order Taylor approximation to message around the point $\hat{r}_i^t$ and the details are described in Appendix C.1. The resulting message then becomes

$$\bar{\mu}_{p(x_i^t \middle| \theta_i^t, s_i) \to \theta_i^t}(\theta_i^t) = \mathcal{N}\left(\theta_i^t \middle| \overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right), \tag{3.19}$$

where the parameters, $\overleftarrow{\xi}_i^t$ and $\overleftarrow{\psi}_i^t$, are obtained from the second order approximation.

The (across) Phase

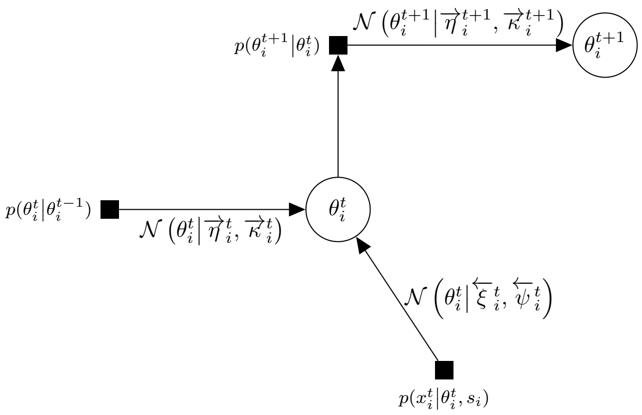

$p(\theta_i^{t+1}|\theta_i^t)$
$\mathcal{N}\left(\theta_i^{t+1}\middle|\overrightarrow{\eta}_i^{t+1}, \overrightarrow{\kappa}_i^{t+1}\right)$
θ_i^{t+1}
$p(\theta_i^t|\theta_i^{t-1})$
θ_i^t
$\mathcal{N}\left(\theta_i^t\middle|\overrightarrow{\eta}_i^t, \overrightarrow{\kappa}_i^t\right)$
$\mathcal{N}\left(\theta_i^t\middle|\overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right)$
$p(x_i^t|\theta_i^t, s_i)$

Figure 3.6: The sub graph for the (across) phase for the variable x_i^t at time t.

The final phase, (across), is responsible to implementing the common sparsity assumption and the temporal correlation of the coefficients. The relevant subgraph is shown in figure 3.6, where we only need to determine the message from the factor node $p(\theta_i^{t+1}|\theta_i^t)$ to variable node θ_i^{t+1}. The remaining messages have already been computed during the previous phases. The sum-product gives the message:

$$\mu_{p(\theta_i^{t+1} \middle| \theta_i^t) \to \theta_i^{t+1}}\left(\theta_i^{t+1}\right) = \int p(\theta_i^{t+1}|\theta_i^t)\mu_{p(\theta_i^t \middle| \theta_i^{t-1}) \to \theta_i^t}\left(\theta_i^t\right)\mu_{p(x_i^t \middle| \theta_i^t, s_i) \to \theta_i^t}\left(\theta_i^t\right) \mathrm{d}\theta_i^t$$
$$= \int \mathcal{N}\left(\theta_i^{t+1}\middle| (1-\alpha)\left(\theta_i^t - \zeta\right) + \zeta, \alpha^2\rho\right)\mathcal{N}\left(\theta_i^t\middle|\overrightarrow{\eta}_i^t, \overrightarrow{\kappa}_i^t\right)\mathcal{N}\left(\theta_i^t\middle|\overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right) \mathrm{d}\theta_i^t$$

where the conditional density $p(\theta_i^{t+1}|\theta_i^t)$ is given in eq. (3.6). The integration is straightforward since all the involved quantities are Gaussian:

$$\mu_{p(\theta_i^{t+1} \middle| \theta_i^t) \to \theta_i^{t+1}}\left(\theta_i^{t+1}\right) = \mathcal{N}\left(\theta_i^{t+1}\middle|\overrightarrow{\eta}_i^{t+1}, \overrightarrow{\kappa}_i^{t+1}\right) \tag{3.20}$$

with

$$\overrightarrow{\eta}_i^{t+1} = (1-\alpha)\left(\frac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \frac{\overleftarrow{\xi}_i^t}{\overleftarrow{\psi}_i^t}\right)\left(\frac{\overrightarrow{\kappa}_i^t \cdot \overleftarrow{\xi}_i^t}{\overrightarrow{\kappa}_i^t + \overleftarrow{\xi}_i^t}\right) + \alpha\zeta \tag{3.21}$$

$$\overrightarrow{\kappa}_i^{t+1} = (1-\alpha)^2\left(\frac{\overrightarrow{\kappa}_i^t \cdot \overleftarrow{\xi}_i^t}{\overrightarrow{\kappa}_i^t + \overleftarrow{\xi}_i^t}\right) + \alpha^2\rho \tag{3.22}$$

This finishes the forward part of the algorithm. Due to the symmetry, the update equations for a backward pass of the algorithm have exactly the form as in the forward pass, except we update the variables $(\overleftarrow{\lambda}, \overleftarrow{\eta}, \overleftarrow{\kappa})$ instead of $(\overrightarrow{\lambda}, \overrightarrow{\eta}, \overrightarrow{\kappa})$ and so on. Hence, we will not spent time deriving the backward part of the algorithm.

Table 3.1 gives an overview over the messages, their parametrization and initialization, while the algorithm for computing a forward pass is summarized in Algorithm 6[3].

Due to the use of the sum-product algorithm, it is possible to obtain estimates of the support variables s_i by computing the posterior distribution of the support conditioned on the observation, i.e. $p(\boldsymbol{s}|\boldsymbol{Y})$. This is easily obtained using the sum-product rules (see table 2.1 in section 2.1):

$$\begin{aligned} p(s_i|\boldsymbol{Y}) &\propto \mu_{p(s_i)\to s_i}(s_i)\prod_{t=1}^{T}\mu_{p(x_i^t|\theta_i^t, s_i)\to s_i}(s_i) \\ &= \text{Ber}(s_i|\lambda)\prod_{t=1}^{T}\text{Ber}(s_i|\overleftarrow{\pi}_i^t) \end{aligned} \tag{3.23}$$

Then a MAP estimate of the support $\boldsymbol{s}$ can be extracted from the above distribution with a minimal amount of computation.

We now consider the computational complexity of the AMP-MMV algorithm. By analyzing the algorithm, it is immediately seen that for a fixed value of $t = \hat{t}$, the phases: (into), (out) and (across) do not scale with m, but they all scale linearly w.r.t. n. Furthermore, the (within) phase has the same computational complexity as BG-AMP, which is $\mathcal{O}(mn)$. The computational complexity per time frame therefore becomes $\mathcal{O}(mn)$. To complete an entire forward pass, we have to repeat the above T times. Therefore, the computational complexity for an entire forward pass then scales as $\mathcal{O}(mnT)$ and performing multiple forward/backward iterations do not change that. Although it does change the

[3]A more compact description of the algorithm is available in [ZS13b]. Notice, the (within) part is stated slightly different in the paper, due to the fact than their derivation is based on [Sch10] rather than [VS13].

Table 3.1: Messages at time step t including their initialization for all 4 phases.

Phase	From	To	Functional form	Initialization
(Into)	$p(s_i)$	s_i	$\text{Ber}(s_i \vert \lambda)$	
	$p(x_i^r \vert \theta_i^r, s_i)$	s_i	$\text{Ber}(s_i \vert \overleftarrow{\pi}_i^r)$	$\overleftarrow{\pi}_i^r = 0.5$
	s_i	$p(x_i^t \vert \theta_i^t, s_i)$	$\text{Ber}(s_i \vert \overrightarrow{\pi}_i^r)$	
	$p(\theta_i^t \vert \theta_i^{t-1})$	θ_i^t	$\mathcal{N}\left(\theta_i^t \vert \overrightarrow{\eta}_i^t, \overrightarrow{\kappa}_i^t\right)$	
	$p(\theta_i^{t+1} \vert \theta_i^t)$	θ_i^t	$\mathcal{N}\left(\theta_i^t \vert \overleftarrow{\eta}_i^t, \overleftarrow{\kappa}_i^t\right)$	$\overleftarrow{\eta}_i^t = 0, \overleftarrow{\kappa}_i^t = \infty$
	θ_i^t	$p(x_i^t \vert \theta_i^t, s_i)$	$\mathcal{N}\left(x_i^t \vert \overrightarrow{\xi}_i^t, \overrightarrow{\psi}_i^t\right)$	
(Within)	BG-AMP			
	x_i^t	$p(x_i^t \vert \theta_i^t, s_i)$	$\mathcal{N}\left(x_i^t \vert \hat{r}_i^t, \tau^r(t)\right)$	
(Out)	$p(x_i^t \vert \theta_i^t, s_i)$	s_i	$\text{Ber}(s_i \vert \overleftarrow{\pi}_i^t)$	
	$p(x_i^t \vert \theta_i^t, s_i)$	θ_i^t	$\mathcal{N}\left(\theta_i^t \vert \overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right)$	
(Across)	$p(\theta_i^{t+1} \vert \theta_i^t)$	θ_i^t	$\mathcal{N}\left(\theta_i^{t+1} \vert \overrightarrow{\eta}_i^{t+1}, \overrightarrow{\kappa}_i^{t+1}\right)$	

Algorithm 6 The forward part AMP-MMV algorithm (AMP-MMV)

```
• For fixed values of hyperparameters $\lambda$, $\sigma^2$, $\zeta$, $\psi$, $\rho$ and $\alpha$.
• **For each** $t = 1..T$, **do**
        (into)-phase. $\forall i \in [n]$:
            $\overrightarrow{\pi}_i^t = \frac{\lambda \prod_{r \neq t} \overleftarrow{\pi}_i^r}{(1-\lambda)\prod_{r \neq t}(1 - \overleftarrow{\pi}_i^r) + \lambda \prod_{r \neq t} \overleftarrow{\pi}_i^r}$
            $\overrightarrow{\xi}_i^t = \left( \frac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \frac{\overleftarrow{\eta}_i^t}{\overleftarrow{\kappa}_i^t} \right) \cdot \overrightarrow{\psi}_i^t, \qquad \overrightarrow{\psi}_i^t = \frac{1}{\frac{1}{\overrightarrow{\kappa}_i^t} + \frac{1}{\overleftarrow{\kappa}_i^t}}$

        (within) phase.
            Use hyperparameters: $\left( \overrightarrow{\pi}_i^t, \overrightarrow{\xi}_i^t, \overrightarrow{\psi}_i^t, \sigma^2 \right)$.
            Initialize BG-AMP as $\hat{x}_i^t = 0, \quad \tau^x(t) = \frac{100}{n} \cdot \sum_{i=1}^{n} \overrightarrow{\psi}_i^t$
            Obtain $(\hat{r}_i^t, \tau^r(t))$, $\forall i \in [n]$ using BG-AMP (scalar variances)

        (out) phase: For each $i \in [n]$:
            $\overleftarrow{\pi}_i^t = \frac{\mathcal{N}\left(0 \middle| \hat{r}_i^t - \overrightarrow{\xi}_i^t, \tau^r(t) + \overrightarrow{\psi}_i^t\right)}{\mathcal{N}\left(0 \middle| \hat{r}_i^t, \tau^r(t)\right) + \mathcal{N}\left(0 \middle| \hat{r}_i^t - \overrightarrow{\xi}_i^t, \tau^r(t) + \overrightarrow{\psi}_i^t\right)}$
            $\left( \overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t \right) = \text{taylor approximation}\left( \overrightarrow{\pi}_i^t, \hat{r}_i^t, \tau^r(t) \right)$

        (across) phase: For each $i \in [n]$:
            $\overrightarrow{\eta}_i^{t+1} = (1-\alpha)\left( \frac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \frac{\overleftarrow{\xi}_i^t}{\overleftarrow{\psi}_i^t} \right)\left( \frac{\overrightarrow{\kappa}_i^t \cdot \overleftarrow{\xi}_i^t}{\overrightarrow{\kappa}_i^t + \overleftarrow{\xi}_i^t} \right) + \alpha\zeta$
            $\overrightarrow{\kappa}_i^{t+1} = (1-\alpha)^2 \left( \frac{\overrightarrow{\kappa}_i^t \cdot \overleftarrow{\xi}_i^t}{\overrightarrow{\kappa}_i^t + \overleftarrow{\xi}_i^t} \right) + \alpha^2\rho$
```

proportionality constant. Therefore, we conclude that the AMP-MMV algorithm scales linearly in all problem dimensions.

Learning the Hyperparameters using EM

In the AMP-MMV algorithm described above, the hyperparameters are considered to be fixed and known in advance. But it is likely that all or some of the hyperparameter requires tuning. Analogous to the EM-BG-AMP algorithm, we will now describe a set of Expectation-Maximization (see section 2.4) update equations for the hyperparameters of the model based on the work in [ZS13b]. We will update the same coordinatewise maximization scheme as used for the BG-AMP model.

Because of the introduction of the hidden variable s_i^t and θ_i^t, we need the posterior distributions $p(s_i^t|\boldsymbol{Y})$ and $p(\theta_i^t|\boldsymbol{Y})$ to compute the E-steps for the hyperparameters for the prior. Fortunately, these are readily obtained using the sum-product rules (see table 2.1 in section 2.1). The posterior distribution of the support is given in eq. (3.23), while the posterior for the coefficients are obtained as follows:

$$\begin{aligned} p(\theta_i^t|\boldsymbol{Y}) &\propto \mu_{p(\theta_i^t|\theta_i^{t-1})\to\theta_i^t}\left(\theta_i^t\right)\mu_{p(\theta_i^{t+1}|\theta_i^t)\to\theta_i^t}\left(\theta_i^t\right)\mu_{p\left(x_i^t|\theta_i^t,s_i\right)\to\theta_i^t}\left(\theta_i^t\right) \\ &= \mathcal{N}\left(\theta_i^t\middle|\overleftarrow{\eta}_i^t,\overleftarrow{\kappa}_i^t\right)\mathcal{N}\left(\theta_i^t\middle|\overrightarrow{\eta}_i^t,\overrightarrow{\kappa}_i^t\right)\mathcal{N}\left(\theta_i^t\middle|\overleftarrow{\xi}_i^t,\overleftarrow{\psi}_i^t\right) \end{aligned} \tag{3.24}$$

Similarly, for estimating the correlation parameter α, we will need the joint posterior $p(\theta_i^t,\theta_i^{t-1}|\boldsymbol{Y})$. But since the two involved variables share a common factor node, this distribution is also easily obtained as the product of the factor function and the incoming messages at the factor node (see table 2.1 in section 2.1):

$$\begin{aligned} p\left(\theta_i^t,\theta_i^{t-1}\middle|\boldsymbol{Y}\right) &\propto p\left(\theta_i^t\middle|\theta_i^{t-1}\right)\cdot\mu_{\theta_i^t\to p\left(\theta_i^t\middle|\theta_i^{t-1}\right)}\left(\theta_i^t\right)\cdot\mu_{\theta_i^{t-1}\to p\left(\theta_i^t\middle|\theta_i^{t-1}\right)}\left(\theta_i^{t-1}\right) \\ &= p\left(\theta_i^t\middle|\theta_i^{t-1}\right)\cdot\mathcal{N}\left(\theta_i^{t-1}\middle|\overrightarrow{\eta}_i^{t-1},\overrightarrow{\kappa}_i^{t-1}\right)\mathcal{N}\left(\theta_i^{t-1}\middle|\overleftarrow{\xi}_i^{t-1},\overleftarrow{\psi}_i^{t-1}\right)\cdot\mathcal{N}\left(\theta_i^t\middle|\overleftarrow{\eta}_i^t,\overleftarrow{\kappa}_i^t\right), \end{aligned} \tag{3.25}$$

where $p\left(\theta_i^t\middle|\theta_i^{t-1}\right)$ is given by eq. (3.6). Note that all quantities involved in the computations of the posteriors are already available, and therefore the E-step is essentially "free" in the context. This really emphasizes the power and flexibility of the message passing approach.

The derivation of the EM-based update equation follows the same approach as described for the EM-BG-AMP model. The details are described in the

Appendix C.2, but here we simply state resulting update equations:

$$\sigma^2_{new} = \frac{1}{Tm} \sum_{t=1}^{T} \sum_{i=1}^{n} \left[(y_a^t - z_a^t)^2 + \tau_a^z \right] \tag{3.26}$$

$$\lambda^{new} = \frac{1}{n} \sum_{i=1}^{n} \frac{\lambda \prod_{t=1}^{T} \overleftarrow{\pi}_i^t}{(1-\lambda) \prod_{t=1}^{T} (1 - \overleftarrow{\pi}_i^t) + \lambda \prod_{t=1}^{T} \overleftarrow{\pi}_i^t} \tag{3.27}$$

$$\zeta^{new} = \left(\frac{(T-1)n}{\rho} + \frac{n}{\sigma_c^2} \right)^{-1} \left(\frac{1}{\sigma_c^2} \sum_{i=1}^{n} \hat{\theta}_i^1 + \frac{1}{\alpha\rho} \sum_{t=2}^{T} \sum_{i=1}^{n} \left[\hat{\theta}_i^t - (1-\alpha)\hat{\theta}_i^{t-1} \right] \right) \tag{3.28}$$

$$\begin{aligned} \rho = \frac{1}{\alpha^2 (T-1) N} \sum_{t=2}^{T} \sum_{i=1}^{n} [\tilde{\theta}_i^t + \left(\hat{\theta}_i^t \right)^2 - 2\zeta\alpha\hat{\theta}^t + (1-\alpha)^2 \left(\tilde{\theta}_i^{t-1} + \hat{\theta}_i^{t-1} \right) + \zeta^2\alpha^2 \\ + 2\zeta\alpha(1-\alpha)\hat{\theta}_i^{t-1} - 2(1-\alpha)\mathbb{E}\left(\theta_i^t \theta_i^{t-1} \middle| \boldsymbol{Y} \right)] \end{aligned} \tag{3.29}$$

$$\alpha^{new} = \frac{b - \sqrt{b^2 + 4N(T-1)c}}{2N(T-1)} \tag{3.30}$$

where

$$b = \frac{1}{\rho} \sum_{t=2}^{T} \sum_{i=1}^{n} \left(\mathbb{E}\left[\theta_i^t \theta_i^{t-1} \middle| \boldsymbol{Y} \right] - \zeta \left(\hat{\theta}_i^t - \hat{\theta}_i^{t-1} \right) - \left(\hat{\theta}_i^{t-1} + \tilde{\theta}_i^{t-1} \right) \right) \tag{3.31}$$

$$c = \frac{1}{\rho} \sum_{t=2}^{T} \sum_{i=1}^{n} \left(\left(\hat{\theta}_i^t + \tilde{\theta}_i^t \right) - 2\mathbb{E}\left[\theta_i^t \theta_i^{t-1} \middle| \boldsymbol{Y} \right] + \left(\hat{\theta}_i^{t-1} + \tilde{\theta}_i^{t-1} \right) \right) \tag{3.32}$$

$$\hat{\theta}_i^t = \tilde{\theta}_i^t \left(\frac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \frac{\overleftarrow{\eta}_i^t}{\overleftarrow{\kappa}_i^t} + \frac{\overleftarrow{\xi}_i^t}{\overleftarrow{\psi}_i^t} \right) \tag{3.33}$$

$$\tilde{\theta}_i^t = \left(\frac{1}{\overrightarrow{\kappa}_i^t} + \frac{1}{\overleftarrow{\kappa}_i^t} + \frac{1}{\overleftarrow{\psi}_i^t} \right)^{-1} \tag{3.34}$$

where $\mathbb{E}\left[\theta_i^t \theta_i^{t-1} \middle| \boldsymbol{Y} \right]$ is obtained from pairwise Gaussian posterior distributions in eq. (3.25).

For multimodal likelihood functions, the initialization of the hyperparameters can have a crucial effect on the results obtained using the EM-algorithm. Therefore, if prior knowledge about the hyperparameter are available, it should be used. Otherwise, initial estimates can be obtained by extracting simple statistic from the measurements $\boldsymbol{Y}$ as described in the EM-BG-AMP algorithm.

Ziniel et al. suggests that the correlation parameter α and the variance parameter ρ are not updated in the same iteration. This is due to the fact that the two parameter are tightly coupled, since the conditional variance of θ_i^t given θ_i^{t-1} is given by:

$$\mathbb{V}\left[\theta_i^t \middle| \theta_i^{t-1} \right] = \alpha^2 \rho \tag{3.35}$$

Therefore, if both α and ρ has been initialized with too small values, the EM algorithm will overcompensate by producing too large values for both parameters and this can lead to oscillatory behaviour [ZS13b]. Furthermore, Ziniel et al. recommend initializing the variance parameters of AMP using: $\tau^x(t) = \frac{100}{n} \cdot \sum_{i=1}^n \overrightarrow{\psi}_i^t$, which also adopt here.

When the AMP-MMV algorithm is used in conjunction with the EM update rules, the resulting algorithm will be denoted EM-AMP-MMV. This algorithm proceeds as follows. Based on the initial values of the hyperparameters, a forward/backward pass is completed. After completing the first pass, the approximate posterior distributions of the random variables s_i and θ_i^s are now available. Using these posterior distributions, the hyperparameters are updated using the EM-scheme. The sequence of a forward/backward pass followed by the EM updates will be referred to as EM or smoothing iterations. This process is then repeated until a maximum number of EM iterations are used or until some stopping criteria are satisfied. For the stopping criteria, we will use:

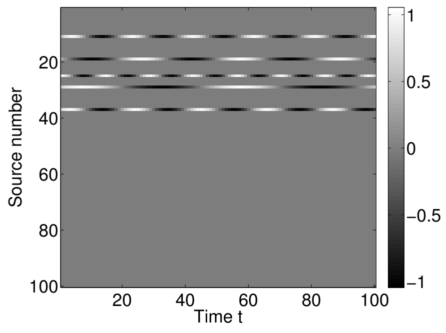
$$\frac{\left|\left|\hat{\boldsymbol{X}}^k - \hat{\boldsymbol{X}}^{k-1}\right|\right|_F^2}{\left|\left|\hat{\boldsymbol{X}}^k\right|\right|_F^2} \leq \tau_{\text{stop}}. \tag{3.36}$$

where $||\cdot||_F$ is the Frobenius norm.

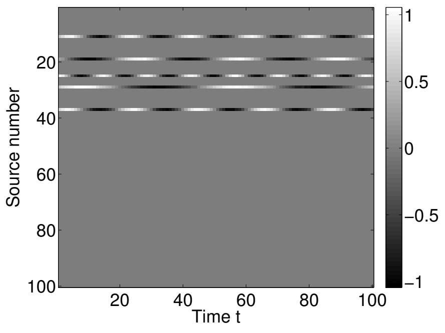
Example: Toy Problem

To illustrate how the EM-AMP-MMV algorithm works, we will now use it to solve a toy MMV problem of the $\boldsymbol{Y} = \boldsymbol{AX} + \boldsymbol{E}$. In this example, the problem size is $n = 100$, the undersamplingsratio is $\delta = 0.15$, the sparsity is $\rho = 0.3$ and the number of measurement vectors is $T = 100$. The forward matrix $\boldsymbol{A}$ is I.I.D. Gaussian, where the columns have been scaled to have unit ℓ_2-norm. The true solution $\boldsymbol{X}$ has 5 non-zero rows, i.e. 5 non-zero sources. These 5 sources are sinusoidal with slightly different frequency. The error matrix $\boldsymbol{E}$ is also I.I.D. Gaussian, where the variance has been scaled to yield a signal-to-noise ratio of SNR$= 20dB$.

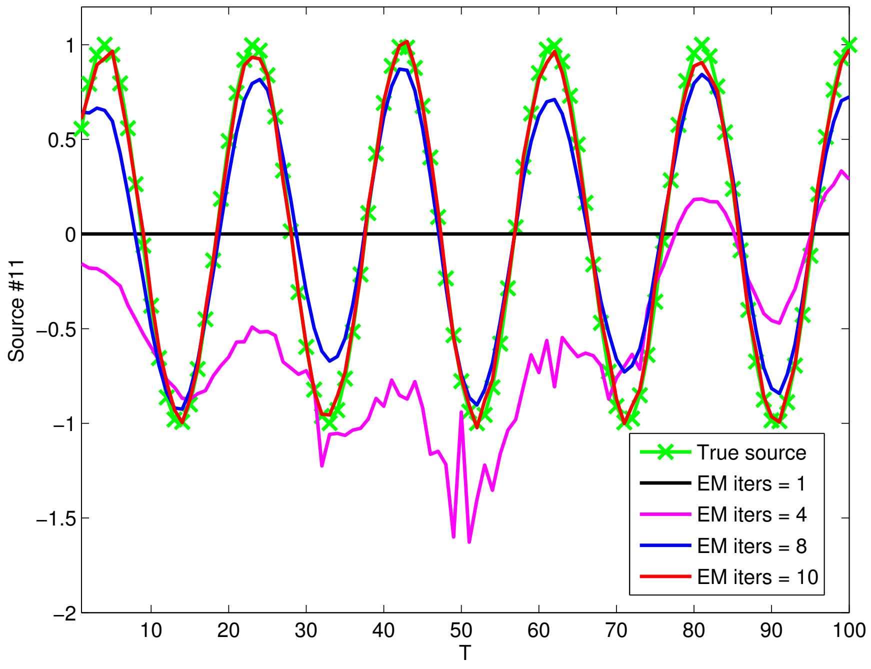
The true sources signal are shown in figure 3.7(a). The figure clearly shows that the true sources have constant support over time. The forward matrix $\boldsymbol{A}$ and the measurement matrix $\boldsymbol{Y}$ are then fed to the EM-AMP-MMV algorithm. The resulting estimate of the sources is shown in figure (b), where it is seen that the algorithm correctly estimates the support. Figure (c) shows how one of the active sources evolves as a function iterations. The green curve is the corresponding true source. It is seen that after the first EM iteration, the estimated signal is simply 0 for values of T(black curve), but after 8 EM iteration the estimated signal has taken the form of a sine wave and after 10 iterations the estimated signal has converged to its final values.



(a) True sources



(b) Estimated sources



(c) Evolution of active source

Figure 3.7: Illustration of the EM-AMP-MMV algorithm. The algorithm is applied to a toy problem with dimension $n = 100$, undersamplingsratio $\delta = 0.15$, sparsity $\rho = 0.3$, SNR $= 20$dB, and then number of measurement vectors are $T = 100$. The true solution $\boldsymbol{X}$ has 5 non-zero sources, which are all sinusoidal with slightly different frequency.

3.2 AMP-DCS: Assuming Dynamic Support

The AMP-MMV algorithm was designed to handle the MMV problem using the common sparsity assumption. The purpose of this section is to relax this assumption and extend the algorithm to handle MMV problems with slowly changing support. We will incorporate the assumed dynamic structure of the support into the prior of AMP-MMV model similar to the work of Ziniel et al. in [ZS13a]. Ziniel et al. suggested this model for the purpose of dynamic compressed sensing and therefore the model is referred to as AMP-DCS.

The common sparsity assumption in the AMP-MMV implies that a given support variable s_i is shared across time for all $\{x_i^t\}_{t=1}^T$. In contrast, we will now introduce an individual support variable s_i^t for each x_i^t and then assume that the support for each source signal, i.e. $\{s_i^t\}_{t=1}^T$ evolve according to a 2-state discrete Markov chain [PK11]. Except for the assumptions on the support, the underlying models for AMP-DCS and AMP-MMV model are identical.

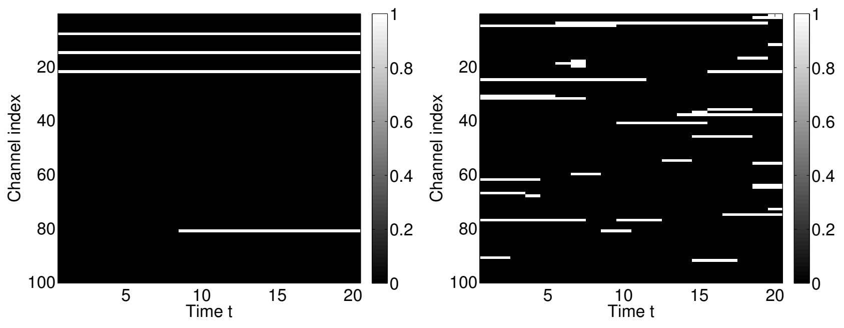
Consider the Markov prior on the support. The support for each row of the solution is assumed to evolve according to an independent, but identical Markov chain. A Markov chain is characterized by an initial probability distribution $p(s_i^1 = 1) = \lambda$ and the transitional probabilities:

$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} \tag{3.37}$$

where, $p_{01} = p(s_i^t = 1 | s_i^{t-1} = 0)$ etc. Since $\boldsymbol{P}$ is a stochastic matrix, each row must sum to 1. We will assume that the Markov chain is operating in steady-state and therefore the Markov chain can be fully characterized by two parameters, which we will denote λ and p_{10}. The parameter λ is the marginal probability of $p(s = 1) = \lambda$ and the latter is the probability of a transition from $s = 1$ to $s = 0$, i.e. $p_{10} = p(s^{t+1} = 0 | s^t = 1)$. The steady state condition also implies[4] $p_{01} = \lambda p_{10}/(1 - \lambda)$. The remaining two diagonal elements p_{00} and p_{11} are determined by the fact that the rows of $\boldsymbol{P}$ sum to 1.

The parameter p_{10} controls how the chain evolves over time. For sufficiently small values of p_{10}, the chain exhibits near static support over time. But when p_{10} increases, the chain will change state more often and thus support becomes "more dynamic". This is illustrated in figure 3.8, which shows two realizations of

[4]The steady-state assumption implies: $\boldsymbol{P}^T \begin{bmatrix} 1-\lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} 1-\lambda \\ \lambda \end{bmatrix}$. The expression for p_{10} is then obtained by solving the eigenvalue problem.



(a) $p_{10} = 0.02$ (b) $p_{10} = 0.2$

Figure 3.8: Realizations of the Markov prior for the support with different values of $p_{10} = p(s_i^t = 0|s_i^{t-1} = 1)$ for $n = 100, \lambda = 0.06, T = 20$

the Markov prior for $\lambda = 0.06$ and for $p_{10} = \{0.2, 0.02\}$. In the left-most figure, it is seen that for the small value of p_{10}, the support is nearly static, whereas the right-most figure shows that the sources "turn on and off" as a result of the larger value of p_{10}. Note, that $1/p_{10}$ is the expected length of a consecutive sequence of ones.

As for the AMP-MMV, the noise is assumed to be I.I.D. Gaussian and the coefficients θ_i^t are assumed to evolve according to the Gauss-Markov process specified in eq. (3.5). Using these assumptions, the joint density for this model is given by

$$p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{S}, \boldsymbol{\theta}) = \prod_{t=1}^{T} p(\boldsymbol{y}^t|\boldsymbol{x}^t)p(\boldsymbol{x}^t|\boldsymbol{s}, \boldsymbol{\theta}^t)p(\boldsymbol{s}^t|\boldsymbol{s}^{t-1})p(\boldsymbol{\theta}^t|\boldsymbol{\theta}^{t-1})$$

$$= \prod_{t=1}^{T}\left(\prod_{a=1}^{m} p(y_a^t|\boldsymbol{x}^t)\prod_{i=1}^{n} p(x_i^t|s_i, \theta_i^t)p(s_i^t|s_i^{t-1})p(\theta_i^t|\theta_i^{t-1})\right) \quad (3.38)$$

where $p(s_i^1|s_i^0) = p(s_i^1) = \lambda$. It is seen that this decomposition is similarly to the corresponding joint density for the AMP-MMV. Hence, the incorporation of the dynamic support does only imply small changes in the underlying factor graph, which is shown in figure 3.9. As before, only the first three time frames are shown, i.e. $t = 1, 2, 3$. Moreover, the dependencies across time for the support and coefficients variables, i.e. s_i^t and θ_i^t, are only shown for the first variable x_1^t to improve visual clarity. But the remaining variables do, of course, have the similar dependencies across time.

Due to the similarity of the underlying factor graphs for the AMP-MMV model and the AMP-DCS model, the majority of the resulting update equations are

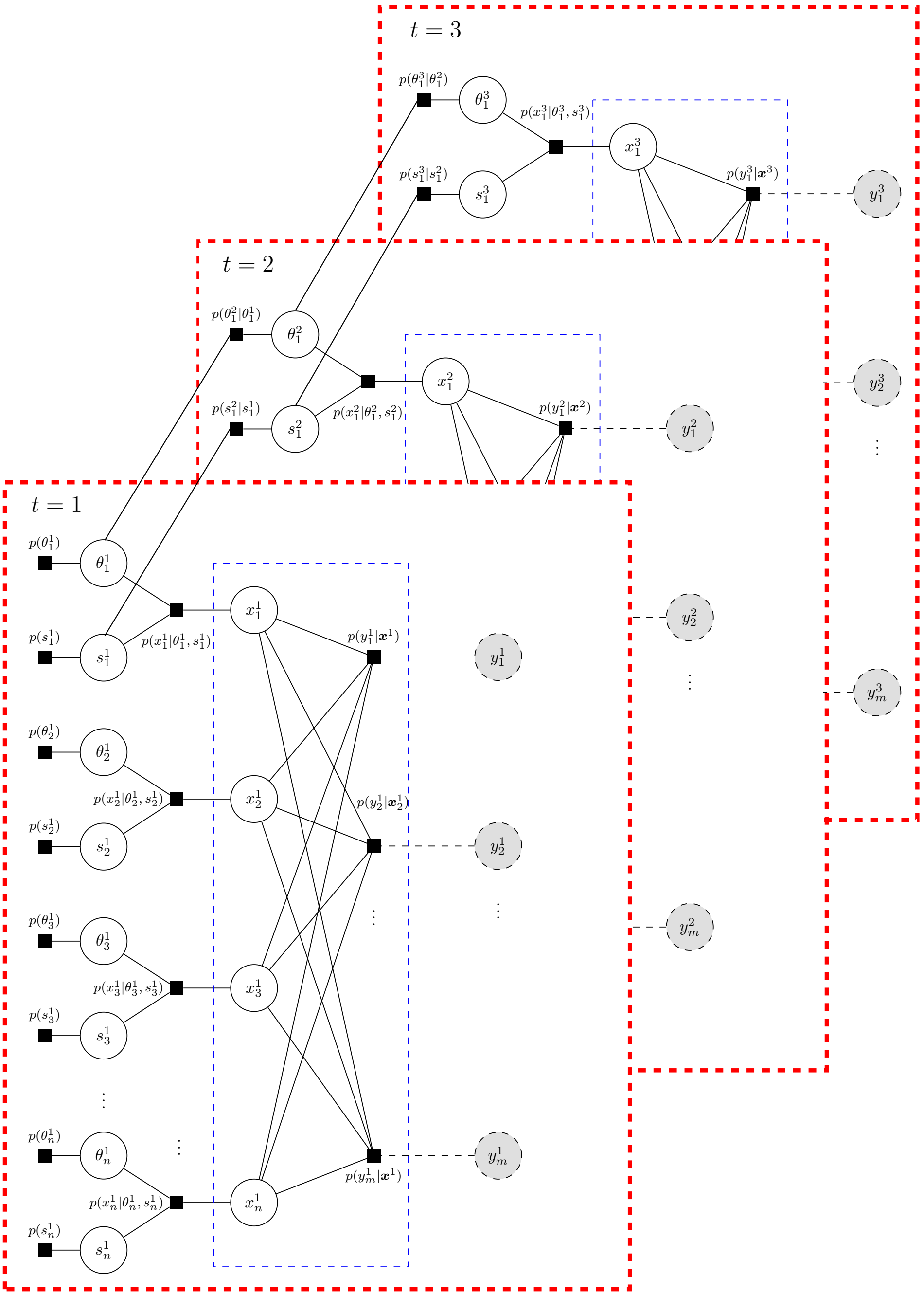


Figure 3.9: The factor graph for the MMV model with dynamic support. The corresponding joint density is given in eq. (3.38). The dependencies across time for the hidden variables, i.e. s_1^t and θ_1^t, are only shown for the first variable x_1 to improve visual clarity. But the remaining variables have the same dependencies.

identical. Therefore, in this section we will only describe the update equations, which are different from the AMP-MMV algorithm. Completely analogous to the AMP-MMV case, the message passing scheme is divided into the four phases. The phases (into), (out) and (across) have minor differences compared to the AMP-MMV case, while the phase (within) is exactly the same as in the AMP-MMV and is therefore not described here. As before, we consider the case for some intermediate t and an arbitrary i in the forward direction.

The (into) Phase for AMP-DCS

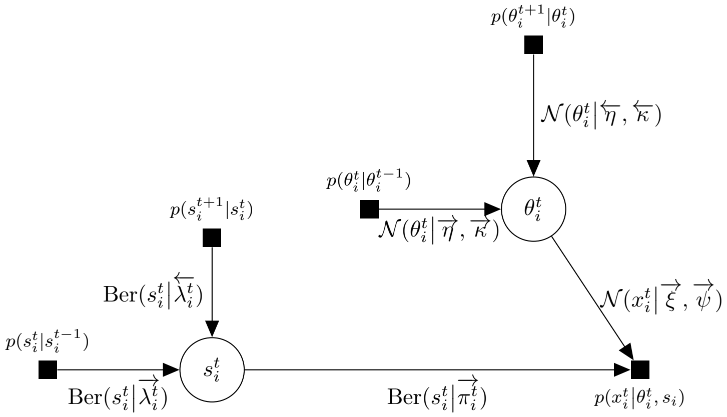
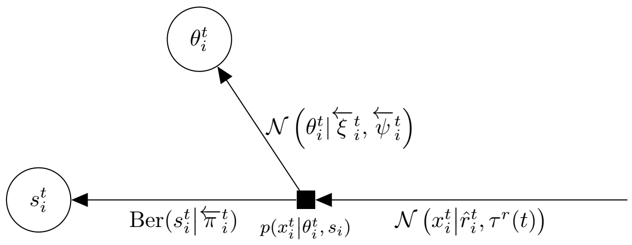

$p(\theta_i^{t+1}|\theta_i^t)$
$\mathcal{N}(\theta_i^t|\overleftarrow{\eta},\overleftarrow{\kappa})$
$p(\theta_i^t|\theta_i^{t-1})$
θ_i^t
$\mathcal{N}(\theta_i^t|\overrightarrow{\eta},\overrightarrow{\kappa})$
$p(s_i^{t+1}|s_i^t)$
$\mathrm{Ber}(s_i^t|\overleftarrow{\lambda}_i^t)$
$\mathcal{N}(x_i^t|\overrightarrow{\xi},\overrightarrow{\psi})$
$p(s_i^t|s_i^{t-1})$
s_i^t
$\mathrm{Ber}(s_i^t|\overrightarrow{\lambda}_i^t)$
$\mathrm{Ber}(s_i^t|\overrightarrow{\pi}_i^t)$
$p(x_i^t|\theta_i^t,s_i)$

Figure 3.10: The sub graph for the (into) phase for the variable x_i^t at time t.

Figure 3.10 shows the factor graph for the (into)-phase for the model with dynamic support. It is seen that each of the support variables s_i^t only are connected to the time frame one step ahead, i.e. $t+1$ and one step behind, i.e. $t-1$. Applying the sum-product rules, the message from variable node s_i^t to factor node $p(x_i^t|\theta_i^t, s_i^t)$ gives rise to the following update equation:

$$\overrightarrow{\pi}_i^t = \frac{\overleftarrow{\lambda}_i^t \cdot \overrightarrow{\lambda}_i^t}{(1-\overleftarrow{\lambda}_i^t)(1-\overrightarrow{\lambda}_i^t) + \overleftarrow{\lambda}_i^t \cdot \overrightarrow{\lambda}_i^t} \tag{3.39}$$

Informally, the belief of s_i^t being active is simply the combined belief of the neighbouring frames. For $t = 1$, $\overrightarrow{\lambda}_i^t$ is initialized as the marginal probability of $s = 1$, i.e. $\overrightarrow{\lambda}_i^1 = \lambda$ for all i.



θ_i^t
$\mathcal{N}\left(\theta_i^t | \overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right)$
s_i^t
$\text{Ber}(s_i^t | \overleftarrow{\pi}_i^t)$
$p(x_i^t | \theta_i^t, s_i)$
$\mathcal{N}\left(x_i^t | \hat{r}_i^t, \tau^r(t)\right)$

Figure 3.11: The sub graph for the (out) phase for the variable x_i^t at time t.

The (out) Phase for AMP-DCS

Figure 3.11 shows the corresponding subgraph for the (out)-phase. It is seen that the topology of the subgraph is exactly the same as for the AMP-MMV case. The only change is that s_i changes to s_i^t. Therefore, the message from factor node $p(x_i^t|\theta_i^t, s_i^t)$ to variable node s_i^t is unchanged. For the message from factor node $p(x_i^t|\theta_i^t, s_i^t)$ to variable node θ_i^t in the AMP-MMV case, we introduced a Taylor approximation due to normalization problems of the true message in eq. (3.16). Based on empirical evidence Ziniel et al. argue that this approach does only perform well for values of $p_{10} < 0.025$ [ZS13a]. To circumvent this, they suggest a simple threshold approximation of the message in eq. (3.17). Thus, for $p_{10} > 0.025$:

$$\left(\overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right) = \begin{cases} \left(\frac{1}{\epsilon}\hat{r}_k^t, \frac{1}{\epsilon^2}\tau^r(t)\right), & \text{if } \overrightarrow{\pi}_i^t \leq \tau \\ \left(\hat{r}_k^t, \tau^r(t)\right), & \text{if } \overrightarrow{\pi}_i^t > \tau \end{cases} \tag{3.40}$$

and for $p_{10} \leq 0.025$, the second order approximation from AMP-MMV case is used.

The (acrosss) Phase for AMP-DCS

As before, this phase is response for implementing the dependencies across time, which in this case is the Markov chain on the support variables and the Gauss-Markov process on the coefficients. Figure 3.12 shows the subgraph for the (across)-phase. The part of the subgraph related to the coefficients is unchanged compared the to AMP-MMV case and therefore the update rule for θ_i^t is identical to those of AMP-MMV. However, the update equation for the probability $\overleftarrow{\pi}_i^t$ does changes, as we will see now. Applying the sum-product rules to the message

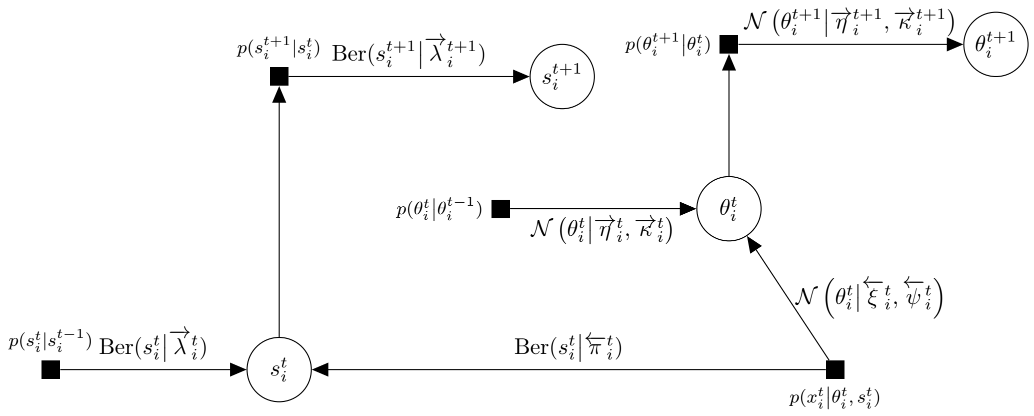

$p(s_i^{t+1}|s_i^t)$
$\mathrm{Ber}(s_i^{t+1}|\overrightarrow{\lambda}_i^{t+1})$
s_i^{t+1}
$p(\theta_i^{t+1}|\theta_i^t)$
$\mathcal{N}\left(\theta_i^{t+1}|\overrightarrow{\eta}_i^{t+1}, \overrightarrow{\kappa}_i^{t+1}\right)$
θ_i^{t+1}
$p(\theta_i^t|\theta_i^{t-1})$
$\mathcal{N}\left(\theta_i^t|\overrightarrow{\eta}_i^t, \overrightarrow{\kappa}_i^t\right)$
θ_i^t
$\mathcal{N}\left(\theta_i^t|\overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right)$
$p(s_i^t|s_i^{t-1})$ $\mathrm{Ber}(s_i^t|\overrightarrow{\lambda}_i^t)$
s_i^t
$\mathrm{Ber}(s_i^t|\overleftarrow{\pi}_i^t)$
$p(x_i^t|\theta_i^t, s_i^t)$

Figure 3.12: The sub graph for the (across) phase for the variable x_i^t at time t.

from factor node $p(s_i^{t+1}|s_i^t)$ to variable node s_i^{t+1} yields:

$$\mu_{p(s_i^{t+1}|s_i^t)\to s_i^{t+1}}(s_i^{t+1}) = \sum_{s_i^t} p(s_i^{t+1}|s_i^t)\mu_{s_i^t\to p(s_i^{t+1}|s_i^t)}(s_i^t)$$

The message from variable node s_i^t to factor node $p(s_i^{t+1}|s_i^t)$ is simply the product of the two incoming messages at node s_i^t and is therefore the product of two Bernoulli distributions. Substituting this into the above expression results in:

$$\mu_{p(s_i^{t+1}|s_i^t)\to s_i^{t+1}}(s_i^{t+1}) \propto \sum_{s_i^t} p(s_i^{t+1}|s_i^t)\left[(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t)(1-s_i^t) + \overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t s_i^t\right] \tag{3.41}$$

Computing the sum over s_i^t using the transitional probabilities $\boldsymbol{P}$ and rearranging gives:

$$\begin{aligned}\mu_{p(s_i^{t+1}|s_i^t)\to s_i^{t+1}}(s_i^{t+1}) &\propto \left[p_{00}(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t) + p_{10}\overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t\right](1-s_i^{t+1}) \\ &\quad + \left[p_{11}\overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t + p_{01}(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t)\right]s_i^{t+1} \\ &= (1-\overrightarrow{\lambda}_i^{t+1})(1-s_i^t) + \overrightarrow{\lambda}_i^{t+1}s_i^t \end{aligned} \tag{3.42}$$

where

$$\overrightarrow{\lambda}_i^{t+1} = \frac{p_{11}\overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t + p_{01}(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t)}{p_{00}(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t) + p_{10}\overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t + p_{11}\overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t + p_{01}(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t)}$$

Using that $p_{00} + p_{01} = 1$ and $p_{10} + p_{11} = 1$, the above reduces to:

$$\overrightarrow{\lambda}_i^{t+1} = \frac{p_{01}(1 - \overleftarrow{\pi}_i^t)(1 - \overrightarrow{\lambda}_i^t) + p_{11}\overleftarrow{\pi}_i^t \overrightarrow{\lambda}_i^t}{(1 - \overleftarrow{\pi}_i^t)(1 - \overrightarrow{\lambda}_i^t) + \overleftarrow{\pi}_i^t \overrightarrow{\lambda}_i^t}$$

This finishes the derivation of the update equations for AMP-DCS method. The forward part of the algorithm is summarized in Algorithm 7. Due to symmetry, the update rules for the backward pass have the same form as for the forward pass, except that we have to update $\overleftarrow{\lambda}_i^{t-1}$ instead of $\overleftarrow{\lambda}_i^{t+1}$ etc.

Learning the Hyperparameters using EM

The EM algorithm described for the AMP-MMV algorithm is also extended to handle the AMP-DCS model. Due to the large similarity of the two models, most of the update equations for the hyperparameters are identical. The only difference is a new update expression for λ and the update expression for the transition probability p_{10}. Here we simply state the update equations, but the details are given in Appendix C.3. The initialization and convergence considerations etc. described for the AMP-MMV model also apply to this model.

The new update equation for the sparsity rate λ is given by:

$$\lambda^{new} = \frac{1}{n}\sum_{i=1}^{n} \frac{\overrightarrow{\lambda}_i^1 \overleftarrow{\lambda}_i^1 \overleftarrow{\pi}_i^1}{(1 - \overrightarrow{\lambda}_i^1)(1 - \overleftarrow{\lambda}_i^1)(1 - \overleftarrow{\pi}_i^1) + \overrightarrow{\lambda}_i^1 \overleftarrow{\lambda}_i^1 \overleftarrow{\pi}_i^1} \tag{3.43}$$

and the update equation for the transition probability p_{10} is given by:

$$p_{10}^{\text{new}} = \frac{\sum_{t=2}^{T}\sum_{i=1}^{n}\mathbb{E}\left[s_i^{t-1}\right] - \sum_{t=2}^{T}\sum_{i=1}^{n}\mathbb{E}\left[s_i^{t-1}s_i^t\right]}{\sum_{t=2}^{T}\sum_{i=1}^{n}\mathbb{E}\left[s_i^{t-1}\right]}, \tag{3.44}$$

where the moments in the latter equations are obtained from the posterior of s_i^t and s_i^{t-1} conditioned on $\boldsymbol{Y}$:

$$p(s_i^t, s_i^{t-1} \big| \boldsymbol{Y}) \propto p(s_i^t|s_i^{t-1}) \cdot \mu_{s_i^{t-1} \to p(s_i^t|s_i^{t-1})}(s_i^{t-1}) \cdot \mu_{s_i^t \to p(s_i^t|s_i^{t-1})}(s_i^t) \tag{3.45}$$

Example: Toy Problem

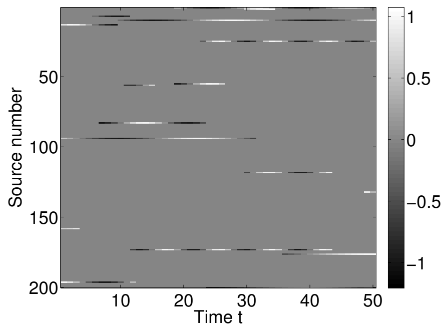
The AMP-DCS algorithm is now illustrated using a simple toy problem of the form $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{E}$. In this example, the problem size is $n = 200$, the undersamplingratio is $\delta = 0.1$, the sparsity is $\rho = 0.3$ and the number of measurement

Algorithm 7 The forward part AMP-DCS algorithm (AMP-DCS)

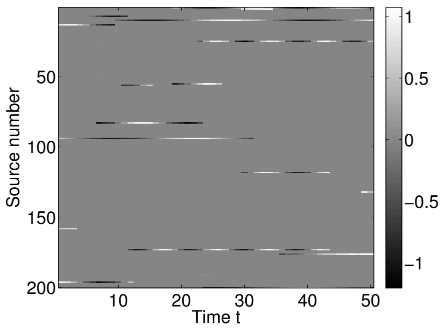
```
• For fixed values of hyperparameters $\lambda$, $p_{10}$, $\sigma^2$, $\zeta$, $\psi$, $\rho$, and $\alpha$.
• **For each** $t = 1..T$, **do**
      (into)-phase. $\forall i \in [n]$:
      $\overrightarrow{\pi}_i^t = \dfrac{\overleftarrow{\lambda}_i^t \cdot \overrightarrow{\lambda}_i^t}{(1-\overleftarrow{\lambda}_i^t)(1-\overrightarrow{\lambda}_i^t) + \overleftarrow{\lambda}_i^t \cdot \overrightarrow{\lambda}_i^t}$
      $\overrightarrow{\xi}_i^t = \left(\dfrac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \dfrac{\overleftarrow{\eta}_i^t}{\overleftarrow{\kappa}_i^t}\right) \cdot \overrightarrow{\psi}_i^t, \qquad \overrightarrow{\psi}_i^t = \dfrac{1}{\frac{1}{\overrightarrow{\kappa}_i^t} + \frac{1}{\overleftarrow{\kappa}_i^t}}$

      (within) phase.
         Use hyperparameters: $\left(\overrightarrow{\pi}_i^t, \overrightarrow{\xi}_i^t, \overrightarrow{\psi}_i^t, \sigma^2\right)$.
         Initialize BG-AMP as $\hat{x}_i^t = 0, \quad \tau^x(t) = 100 \cdot \sum_{i=1}^{n} \overrightarrow{\psi}_i^t$
         Obtain $\left(\hat{r}_i^t, \tau^r(t)\right)$, $\forall i \in [n]$ using BG-AMP (scalar variances)

      (out) phase: For each $i \in [n]$:
      $\overleftarrow{\pi}_i^t = \dfrac{\mathcal{N}\left(0 | \hat{r}_i^t - \overrightarrow{\xi}_i^t, \tau^r(t) + \overrightarrow{\psi}_i^t\right)}{\mathcal{N}\left(0 | \hat{r}_i^t, \tau^r(t)\right) + \mathcal{N}\left(0 | \hat{r}_i^t - \overrightarrow{\xi}_i^t, \tau^r(t) + \overrightarrow{\psi}_i^t\right)}$
      $\left(\overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right) =$ taylor approximation $\left(\overrightarrow{\pi}_i^t, \hat{r}_i^t, \tau^r(t)\right)$

      (out) phase: For each $i \in [n]$:
      $\overleftarrow{\pi}_i^t = \dfrac{\mathcal{N}\left(0 | \hat{r}_i^t - \overrightarrow{\xi}_i^t, \tau^r(t) + \overrightarrow{\psi}_i^t\right)}{\mathcal{N}\left(0 | \hat{r}_i^t, \tau^r(t)\right) + \mathcal{N}\left(0 | \hat{r}_i^t - \overrightarrow{\xi}_i^t, \tau^r(t) + \overrightarrow{\psi}_i^t\right)}$
         if $p_{10} \leq 0.025$
            $\left(\overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right) =$ taylor approximation $\left(\overrightarrow{\pi}_i^t, \hat{r}_i^t, \tau^r(t)\right)$
         else
            $\left(\overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right) = \begin{cases} \left(\frac{1}{\epsilon}\hat{r}_k^t, \frac{1}{\epsilon^2}\tau^r(t)\right), & \text{if } \overrightarrow{\pi}_i^t \leq \tau \\ \left(\hat{r}_k^t, \tau^r(t)\right), & \text{if } \overrightarrow{\pi}_i^t > \tau \end{cases}$

      (across) phase: For each $i \in [n]$:
      $\overrightarrow{\eta}_i^{t+1} = (1-\alpha)\left(\dfrac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \dfrac{\overleftarrow{\xi}_i^t}{\overleftarrow{\psi}_i^t}\right)\left(\dfrac{\overrightarrow{\kappa}_i^t \cdot \overleftarrow{\xi}_i^t}{\overrightarrow{\kappa}_i^t + \overleftarrow{\xi}_i^t}\right) + \alpha\zeta$
      $\overrightarrow{\kappa}_i^{t+1} = (1-\alpha)^2\left(\dfrac{\overrightarrow{\kappa}_i^t \cdot \overleftarrow{\xi}_i^t}{\overrightarrow{\kappa}_i^t + \overleftarrow{\xi}_i^t}\right) + \alpha^2\rho$
      $\overrightarrow{\lambda}_i^{t+1} = \dfrac{p_{01}(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t) + p_{11}\overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t}{(1-\overleftarrow{\pi}_i^t)(1-\overrightarrow{\lambda}_i^t) + \overleftarrow{\pi}_i^t\overrightarrow{\lambda}_i^t}$
```

vectors is $T = 50$. The forward matrix $\boldsymbol{A}$ is I.I.D. Gaussian, where the columns have been scaled to have unit ℓ_2-norm. The true solution $\boldsymbol{X} = \boldsymbol{S}\bar{\boldsymbol{\theta}}$ is generated as follows. The support $\boldsymbol{S}$ is sampled from the Markov prior with the hyperparameters $\lambda = \delta\rho$ and $p_{10} = 1/20$. This implies that average number of active sources is $\lambda \cdot n = 6$. The coefficients $\bar{\boldsymbol{\theta}}$ are sine waves. That is, each row of $\bar{\boldsymbol{\theta}}$ corresponding sinusoidal signal with slightly frequency. The resulting solution matrix $\boldsymbol{X}$ is shown in figure 3.13(a), where it is clearly that the common sparsity assumption is violated. The error matrix $\boldsymbol{E}$ is I.I.D. Gaussian, where the variance has been scaled to yield a signal-to-noise ratio of SNR= $20dB$. The EM-AMP-DCS algorithm is then used to reconstruct $\boldsymbol{X}$ from $\boldsymbol{A}$ and $\boldsymbol{Y}$.

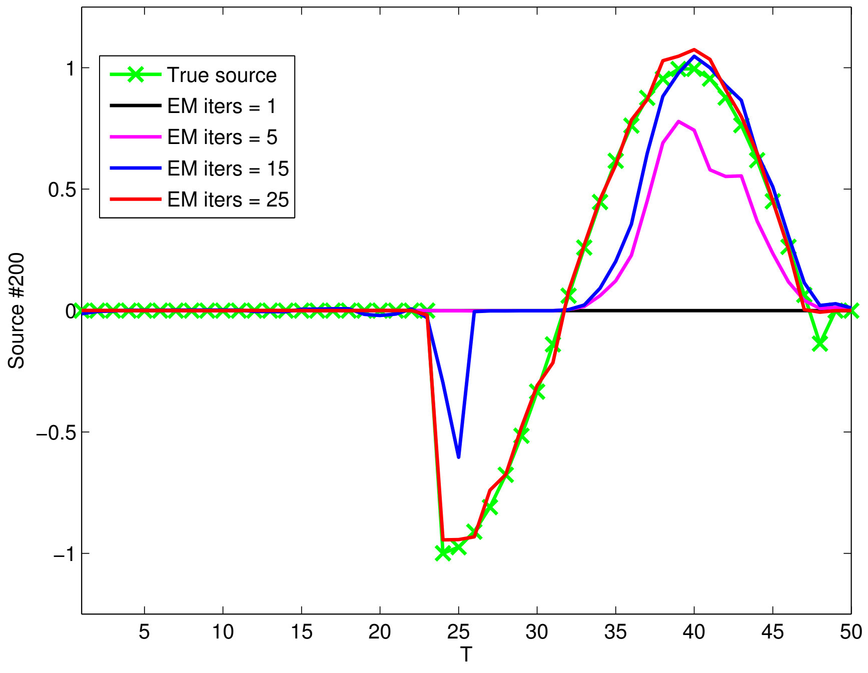
The estimated solution $\hat{X}$ is shown in figure 3.13(b), where it is seen that the AMP-DCS method is capable of estimating sources, which are only active in a proportion of the signal. By comparing the figure (a) and (b), the estimated support is very close to the true support. Figure (c) shows the evolution of 200th source as a function of the number of EM iterations superimposed with the true source. It is seen that the estimated sources is zero after the first iteration (black curve), but then it gradually approaches to the true source and after 25 iterations the estimated source is pretty close to the true source. Although it fails to detect the small negative peak at $t = 48$.



(a) True sources



(b) Estimated sources



(c) Evolution of arbitrary active source

Figure 3.13: Illustration of the EM-AMP-DCS algorithm using a toy problem. The problem is generated using $n = 200, \delta = 0.1, \rho = 0.3, \lambda = \delta\rho, T = 50, \zeta = 0, \psi = 1, T = 50, p_{10} = 1/20, \alpha = 0.9$ and solved using EM-DCS-AMP. (a) The true sources (b) Estimated sources (c) Evolution of an arbitrary active source as a function of EM iterations.

CHAPTER 4

Numerical Experiments

In order to analyze the performance and the properties of the algorithms described in chapter 2 and chapter 3, a series of numerical experiments have been designed and conducted.

This chapter is divided into six subsections. The first three subsections are devoted to investigate the algorithms for the single measurement problem (SMV). Specifically, in section 4.1 a number of small experiments are set up to examine the properties of the AMP0 algorithm. Section 4.2 describes an experiment, which is used to determine the empirical phase transition curves for AMP0 and EM-BG-AMP for noiseless problems. Section 4.3 considers noisy problems and compares the EM-BG-AMP algorithm to other reconstruction algorithms.

The last three subsections addresses algorithms for the MMV formulation, where sections 4.4 and 4.5 consider the AMP-MMV and AMP-DCS algorithms, respectively. Finally, section 4.6 describes a simulation, which is designed the mimic the true properties of an EEG inverse problem.

Recall, that for a linear inverse problem of the form $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$, the undersamplingratio is defined as $\delta = m/n$ and the sparsity is defined as $\rho = k/m$, where k is the true number of non-zero elements in $\boldsymbol{x}$. We will make extensive use of these definitions throughout this chapter. All experiments are based on synthetic data and all simulations are performed in Matlab R2013a 64 bit.

4.1 Analysis of the AMP Algorithm

This section is devoted to the explore the properties of the AMP-algorithm, which was derived in section 2.2.

Performance vs. Problem Size

The AMP algorithm was derived using the large system limit, i.e. $n, m \to \infty$ for $m/n \to \delta$. It is therefore of particular interest to investigate the performance of the algorithm as a function of the problem size n. If the method only is able to recover the solution for extremely large systems, e.g. $n > 10^7$, the applicability to practical problems will be limited. Although, the asymptotic properties are still of theoretical interest.

The first experiment is therefore designed to investigate the significance of the problem size n in a noiseless setting. That is, in this experiment the algorithm AMP0 is applied to a series of problems with different size n. To quantify the performance of the algorithm, we will define the criteria of success in terms the relative error:

$$\frac{||\boldsymbol{x}_0 - \hat{\boldsymbol{x}}||_2}{||\boldsymbol{x}_0||_2} \leq \tau_{\text{success}}, \tag{4.1}$$

where $\boldsymbol{x}_0$ is the true solution, $\hat{\boldsymbol{x}}$ is the estimated solution and τ_{success} is a threshold parameter. That is, we say that the specific problem instance has been successfully solved, if the relative error of estimate solution is smaller than the threshold τ_{success}. This criteria is meaningful, since the experiment is conducted in a noiseless setting. For the threshold parameter we will use $\tau_{\text{success}} = 10^{-2}$ as in [MD10] to facilitate comparison.

Define the success variable $\mathcal{S}_r \in \{0, 1\}$ as $\mathcal{S}_r = 1$ if and only if the r'th run is a success. The average of $\{\mathcal{S}_r\}_{r=1}^{R}$ can then be interpreted as the empirical probability of success.

We will therefore measure the empirical probability of success as a function of the problem size n. The experiment is conducted as follows. The degree of sparsity is fixed to $\rho = \frac{1}{20}$. Then for a specific set of values for n and δ, the forward matrix is generated by sampling the elements A_{ai} from an I.I.D. Gaussian distribution and then scaling the columns of $\boldsymbol{A}$ to unit ℓ_2-norm. The number of non-zero components in the true solution $\boldsymbol{x}_0$ is fixed to the nearest integer to $k = \rho \cdot \delta \cdot n$ and the coefficients of all non-zero elements are fixed to 1 for simplicity. The measurements for each problem instances $\boldsymbol{y}$ are then generated using $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$.

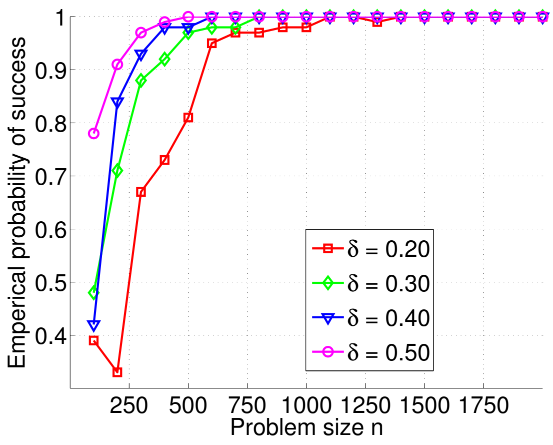


Figure 4.1: Performance vs. problem size for AMP0 in a noiseless setting. The problems are generated using Gaussian I.I.D. forward matrices, where the columns have been scaled to unit ℓ_2-norm. The sparsity is fixed to $\rho = \frac{1}{20}$, and thus the number of non-zero elements in $\boldsymbol{x}_0$ is fixed to nearest integer to $k = \rho\delta n$. The coefficients of all non-zero elements are chosen to 1. The measurements for each problem instance are then generated using $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$ and recovered using AMP0. The results are averaged over $R = 100$ runs.

For 4 different values of δ, we then sweep over the problem size n in the interval $[100, 2000]$ with a step size of 100. For each value of n, the AMP0 algorithm is then applied to the resulting problem. The maximum number of allowed iterations is fixed to 500 with the possibility of early stopping, if

$$\frac{\left|\left|\hat{\boldsymbol{x}}^k - \hat{\boldsymbol{x}}^{k-1}\right|\right|_2}{\left|\left|\boldsymbol{x}^k\right|\right|_2} \leq 10^{-6} \tag{4.2}$$

where k is the iteration number. This process is repeated $R = 100$ times for each value of n and δ.

Figure 4.1 shows the estimated probabilities of success as a function of n. As expected a clear dependency on the problem size n as well as δ is seen. For the highest undersamplingsratio, i.e. $\delta = 0.5$, the problem size has to be at least $n = 500$ for perfect recovery, while for the lowest undersamplingsratio, i.e. $\delta = 0.2$, the problem size n has to be larger than 1250. The dependencies on both n and δ are expected since derivation of AMP0 is based on approximations, whose quality depends on both n and m. Despite this dependency, it is worth noting that the AMP0 algorithm provides perfect recovery at the computational cost $\mathcal{O}(mn)$ for even small/medium scale systems as small as $n = 500$. This makes the AMP-framework very applicable to medium and large scale problems since the approximations are only expected to become better as the problem size increases.

Performance of AMP vs. MP

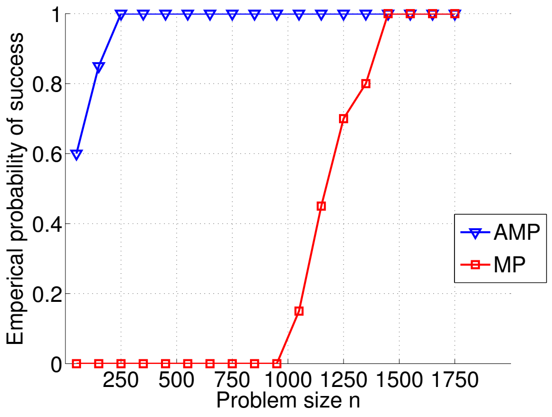
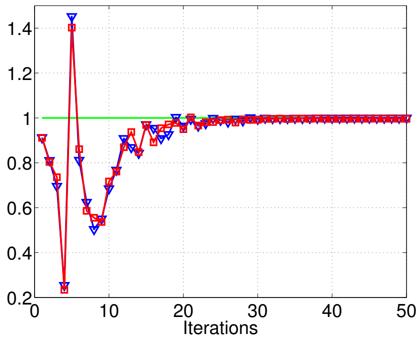


Figure 4.2: Performance vs. problem size for AMP0 and MP in a noiseless setting. The problems are generated using Gaussian I.I.D. forward matrices, where the columns have been scaled to unit ℓ_2-norm. The undersamplings rati is fixed to 0.5 and the sparsity is fixed to $\rho = \frac{1}{20}$, and thus the number of non-zero elements in $\boldsymbol{x}_0$ is fixed to nearest integer to $k = \rho\delta n$. The coefficients of all non-zero elements are fixed to 1. The measurements is then generated as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$ and recovered using AMP0 and MP. The results are averaged over $R = 20$ runs.

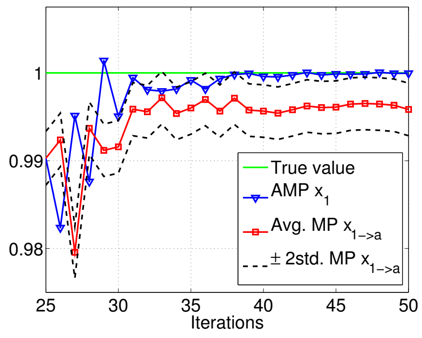
In the derivation of the AMP-algorithm, a series of approximations were applied to a set of message passing update equations. In order to gain further insight into this algorithm, the next experiment is designed to examine the properties of the last approximation, in which the number of messages was reduced from $2mn$ to $m + n$. The message passing (MP) algorithm with $2mn$ messages is given in eq. (2.69) and eq. (2.70) and will be referred to as MP (in contrast to AMP) in the following.

Since the AMP scheme is an approximation to the MP scheme, it is natural to expect better performance from the MP scheme, but at the cost of higher computational complexity. First, we compare to the two schemes using an experiment similar to the one conducted in the previous section. However, δ is fixed to $\delta = 0.5$ and the results are averaged over $R = 20$ runs.

The result is shown in figure 4.1. It is seen that the MP scheme seems to require a much larger problem size than AMP0 in order to achieve perfect recovery. Although not conclusive, this result suggests that the approximation of the MP scheme into the AMP scheme has a positive effect on the sensitivity to the problem size n. This is perhaps due to the fact that the MP scheme has a much higher number of parameters to estimated in each iteration compared to the AMP scheme.



(a)



(b)

Figure 4.3: Evolution of the AMP and MP messages sent from the first variable x_1 to the factor nodes. The data are generated from a noiseless problem with parameters $n = 2000, \delta = 0.5$ and $\rho = 1/20$. The coefficients of the non-zero weights are fixed to 1.

Recall that the messages from variable nodes to factor nodes in the AMP scheme are independent of the index of the factor nodes, whereas the corresponding messages in the MP scheme are indeed dependent on the index of the factor nodes. We will now examine how the messages of two algorithms evolve as a function of iterations. In particular, we will compare the evolution of the messages sent from the first variable node, i.e. x_1 in both AMP and MP. Since the messages in the MP scheme depends on the factor node index, the average message will be used. That is, we compare x_1^{AMP} to $\frac{1}{m}\sum_{a=1}^{m} x_{i\to a}^{\text{MP}}$. The parameters of the problem are chosen to $\delta = 0.5$, $\rho = 0.05$ for $n = 2000$ such that MP algorithm actually converges, cf. figure 4.2.

The result is shown in figure 4.3(a)-(b). The values of both messages fluctuates heavily during the first 25 iterations, after which they both converge to a value

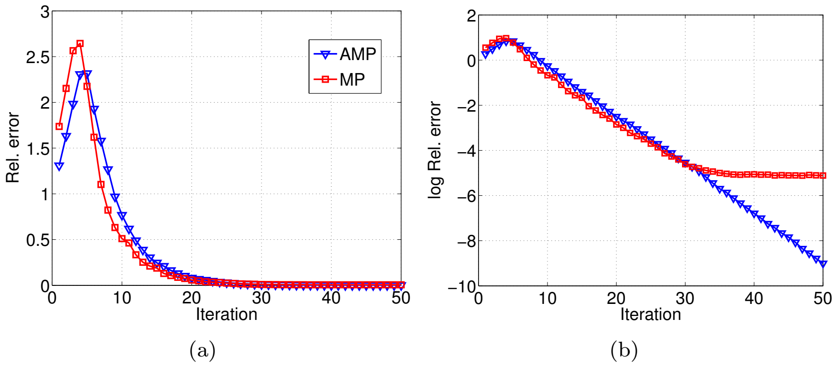


Figure 4.4: Evolution of relative errors for the AMP and MP schemes. The data are generated from a noiseless problem with parameters $n = 2000, \delta = 0.5$ and $\rho = 1/20$. The coefficients of the non-zero weights are fixed to 1.

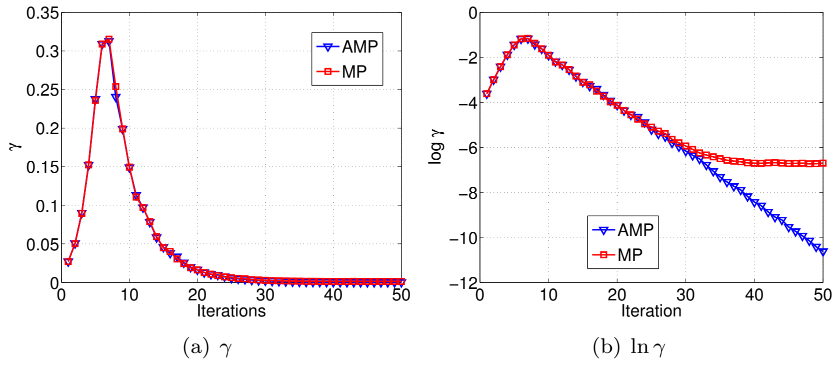


Figure 4.5: Evolution of threshold parameter γ for the AMP and MP scheme. The data are generated from a noiseless problem with parameters $n = 2000, \delta = 0.5$ and $\rho = 1/20$. The coefficients of the non-zero weights are fixed to 1.

close to the true value. Figure 4.3b zooms in on the region of interest for the last 25 iterations. This figure also depicts the variation within the MP messages by plotting ± 2 standard deviations of the messages (black dashed line). Note, that the values of the AMP messages (blue curve) are actually closer to the true value (green line) than the MP messages (red curve).

Figure 4.4(a) shows the relative error as a function of iterations for the two methods. The two curves have very similar shapes and both seem to converge after approximate 30 iterations. However, figure 4.4(b) shows the logarithm of the same curves and here it is seen that the MP method actually reaches a plateau, while the AMP curve decreases steadily. A similar phenomena is observed on figure 4.5(a)-(b), which shows the evolution of the threshold parameter for the AMP and MP schemes. In the first 25 iterations, the two curves are almost inseparable, but after approximately 30 iterations the threshold parameter for MP reaches a plateau. These observations are consistent with the observations from figure 4.3(b).

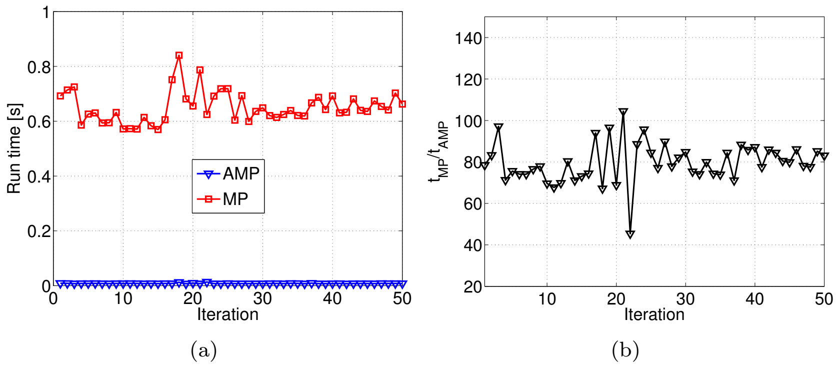


Figure 4.6: Run time for AMP and MP as a function of iterations. The data are generated from a noiseless problem with parameters $n = 2000, \delta = 0.5$ and $\rho = 1/20$. The coefficients of the non-zero weights are fixed to 1.

Finally, figure 4.6(a) shows the run time in seconds as a function of iterations for both methods and figure 4.6(b) shows the ratio of the two run times. Of course, the observed run times are heavily dependent on the specific implementation and the specific hardware details. Therefore, we are not interested in the absolute values of the run times, but only the relative values. The left-most figure shows that the average run time for MP is approximately 80 times higher than the run time of AMP.

Thus, this little experiment suggests that the approximation, in which the number of messages is reduced from $2mn$ to $m+n$, does not degrade the performance of the algorithm significantly. In fact, for this particular type of problem the recovery performance is increased, while giving a significant reduction in run time. However, for larger problems the recovery performance of MP is expected to be equal to or better than AMP, but at the cost of significantly higher computational complexity.

4.2 Phase Transition Curves

In general, the difficulty of a linear inverse problem increases when the number of samples decrease or when the number of non-zero elements increase. Hence, it is of great interest to develop methods, which have a good performance in terms of the sparsity and under-sampling trade-off. As stated earlier (see Literature review section 1.2), many methods for solving linear inverse problems exhibit a sharp phase transition, which effectively partitions the (δ, ρ)-plane into an unsolvable and a solvable region. The boundary between these two regions, i.e. the phase transition curve, then provides a principled way of comparing performance in terms of the undersampling sparsity trade-off.

We will now estimate the empirical phase transition curve for AMP0 and compare it to EM-BG-AMP, Bayesian Variational Garrote (VG) [AHH13], and FOCUSS [GR97] methods. The minimum ℓ_2 norm estimated is used as the initial estimate for the FOCUSS method. Although, the methods VG and EM-BG-AMP are not designed for the noiseless setting, they are included in the comparison anyway since they do not require any parameter tuning[1]. In real life applications, the measurements are often heavily contaminated with noise, but it is still interesting to investigate the performance in the noiseless case since it provides a bound on the achievable performance. That is, if a given problem is unsolvable in the noiseless setting, we should not hope to be able to solve it in the noisy setting.

The phase transition curve is estimated using the same approach as described in [MD10] and the procedure is as follows. The problem size is fixed to $n = 500$, while the sparsity ρ and the undersamplingsratio δ are sampled equidistant in the interval $[0.05, 0.95]$ in steps of 0.05. For each (δ, ρ)-pair, $R = 50$ problems are generated and then attempted solved using the method under examination. Each problem instance is generated using an I.I.D. Gaussian forward matrix $\boldsymbol{A}$ of suitable size, where the columns have been scaled to unit ℓ_2-norm. The number of non-zero elements in the true solution $\boldsymbol{x}_0$ is set to the nearest integer

[1]The original Variational Garrote [KG12] requires tuning of the sparsity controlling parameter, but the Bayesian extension used here learns the degree of sparsity from the data.

to $k = \delta \cdot \rho \cdot n$ and the coefficients of non-zero entries are fixed to 1. Noiseless measurements are then generated using $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0$. If the estimated solution of the r'th problem instance satisfies the criteria in eq. (4.1), we set $\mathcal{S}_r(\delta, \rho) = 1$. Otherwise, we set $\mathcal{S}_r(\delta, \rho) = 0$. For each (δ, ρ)-pair, we can now define the

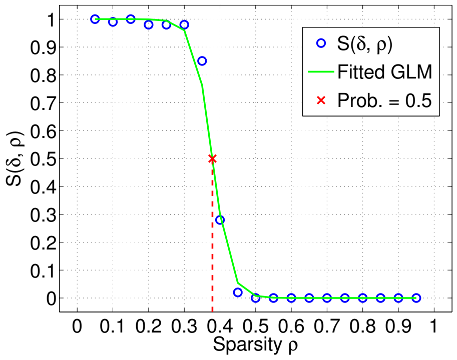


Figure 4.7: Example of how to estimate the phase transition point for a specific value of δ. The blue points corresponds to the estimated probabilities of success. A generalized linear model is then fitted to these points (green curve) and the value of ρ, for which the probability is equal to 0.5 is said to be the phase transition point $\rho^*(\delta)$.

empirical probability of success as

$$\mathcal{S}(\delta, \rho) = \frac{1}{R} \sum_{r=1}^{R} \mathcal{S}_r(\delta, \rho) \tag{4.3}$$

For a specific value of $\delta = \hat{\delta}$, we can now consider $\mathcal{S}(\hat{\delta}, \rho)$ as a function of ρ, and then fit a generalized linear model [MT11] of the form:

$$\text{logit}\left[\mathcal{S}(\hat{\delta}, \rho)\right] = a + b\rho \tag{4.4}$$

Then the *finite-n* phase transition point $\rho^*(\hat{\delta})$ is defined as the value of ρ, for which the probability of success is equal to 50%. Using the GLM model, this point is easily computed as $\rho^*\left(\hat{\delta}\right) = -a/b$. Figure 4.7 shows an example of this for a specific value of δ. The blue points correspond to the empirical probabilities and the green curve is the fitted GLM model. The red cross then indicates the point, $\rho^*(\hat{\delta}) = 0.38$, where the probability is equal to 50% according to the model. This process is then repeated for all values of δ and then $\rho^*(\delta)$ corresponds to the empirical phase transition curve.

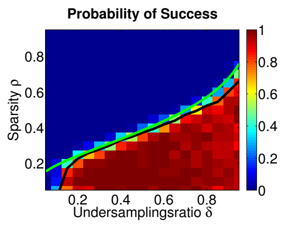
Note, that the choice of problem size, i.e. $n = 500$, may cause trouble for the AMP0 method for low values of δ. But this is a necessary trade-off, since these kinds of experiments are indeed computational demanding and very time consuming[2].

The maximum number of allowed iterations for AMP0, EM-BG-AMP, FOCUSS, and VG Bayes are fixed to $500, 60, 60, 500$, respectively, which are well above the necessary number of iterations for all methods. All with the possibility of early stopping, if the criteria in eq. (4.2) is satisfied. The actual number of iterations used and CPU run time are also measured for each (δ, ρ)-pair for each methods. The results are shown in figures 4.8(a)-(l). The first column shows the estimated probability of success for each (δ, ρ)-pair superimposed with the estimated phase transition (dashed black line) and the theoretical phase transition curve for ℓ_1 minimization (green).

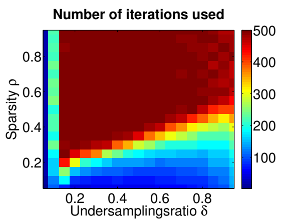
By inspecting the left-most column, it is seen that all methods exhibit a sharp phase transition. Clearly, the EM-BG-AMP provides the best phase transition curve. A closer look on the figure (a), reveals that there is some ”ripple” in the ”solvable” region for the AMP0 method. This may be caused by the fact that the problem size is only $n = 500$. In figure (b), it appears that the number of iterations is approximately constant for sufficiently low values of ρ, but increases when ρ approaches the phase transition point. In the unsolvable region above the curve, it simply spends the maximum number of iterations. The same pattern is reflected in figure (c), which shows the CPU run time in seconds. But note the scale of figure (c) compared to (f),(i), and (l). AMP0 is magnitudes faster than VG and FOCUSS. Furthermore, it is worth mentioning that even if the EM-BG-AMP method is running the EM update scheme, it is still much faster than both FOCUSS and VG. Notice also the remarkable small number of iterations used for the FOCUSS methods as shown in figure (h).

The four estimated phase transition curves are all shown in figure 4.9 to facilitate comparison. The estimated phase transition curve for AMP0 is close to the theoretical curve for the ℓ_1 minimization approach as expected. However, for $\delta < 0.1$ the curve drops significantly compared to the theoretical curve. This is a property of the thresholding policy used in the AMP0 method. The same phenomena is seen in [DMM10] and is confirmed in [Mal13]. Donoho et al. have developed another thresholding policy, which does not suffer from this issue [DMM09]. Instead of using τ^k or γ^k as threshold parameter in the k'th iteration, they use $\lambda\sigma_k$ as threshold parameter, where λ is tuning parameter and σ_k is an estimate of mean square error of the current estimate $\boldsymbol{x}^k$. When λ is optimally tuned, this approach is referred to as *Optimally tuned AMP* [DMM09].

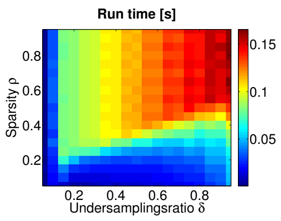
[2]For $R = 50$ and the specific resolution of δ and ρ, we have to solve $19 \cdot 19 \cdot 50 = 18050$ inverse problems to estimate the phase transition curve for one of the methods.



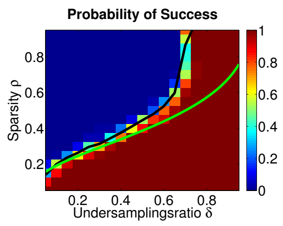
(a) AMP0



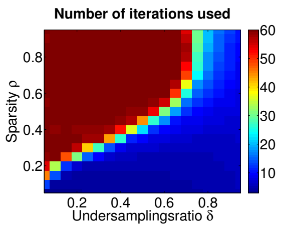
(b) AMP0



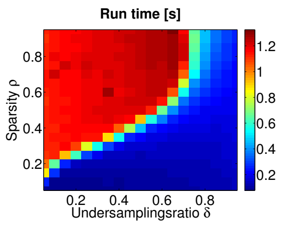
(c) AMP0



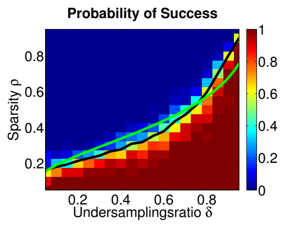
(d) EM-BG-AMP



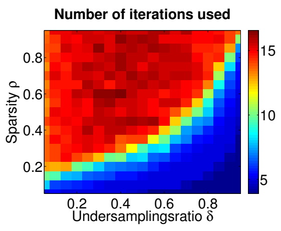
(e) EM-BG-AMP



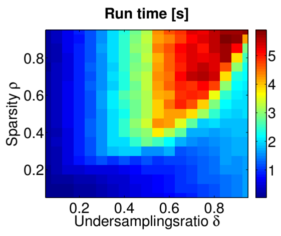
(f) EM-BG-AMP



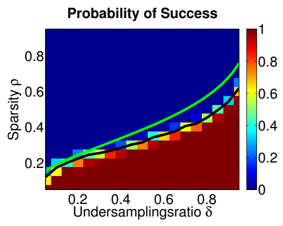
(g) FOCUSS



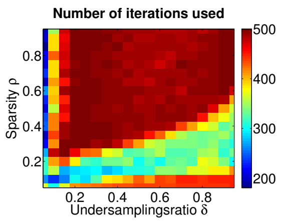
(h) FOCUSS



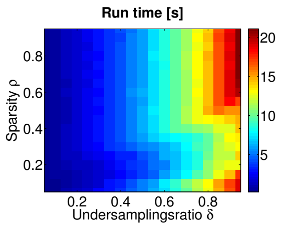
(i) FOCUSS



(j) VG Bayes



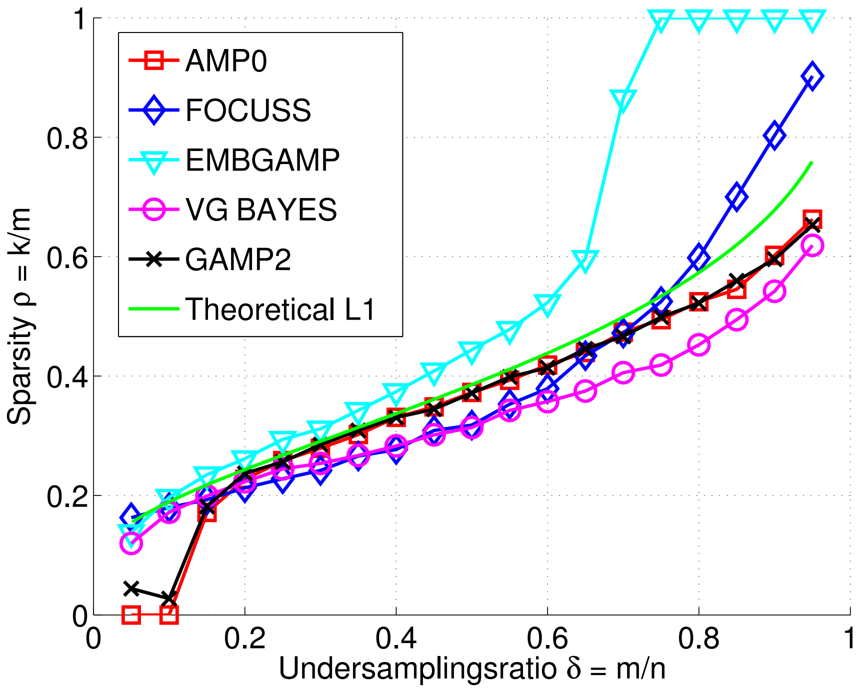
(k) VG Bayes



(l) VG Bayes

Figure 4.8: Comparison of the methods AMP0, EM-BG-AMP, FOCUSS and VG. Data are generated using $n = 500$, Gaussian I.I.D forward matrix, and the coefficients of all non-zero weights are fixed to 1. The values of δ and ρ are both sampled equidistant in the interval $[0.05; 0.95]$. The results are averaged over $R = 50$ runs. Left column: Estimated probability of success. Mid column: Number of iterations. Right column: Run time in seconds.

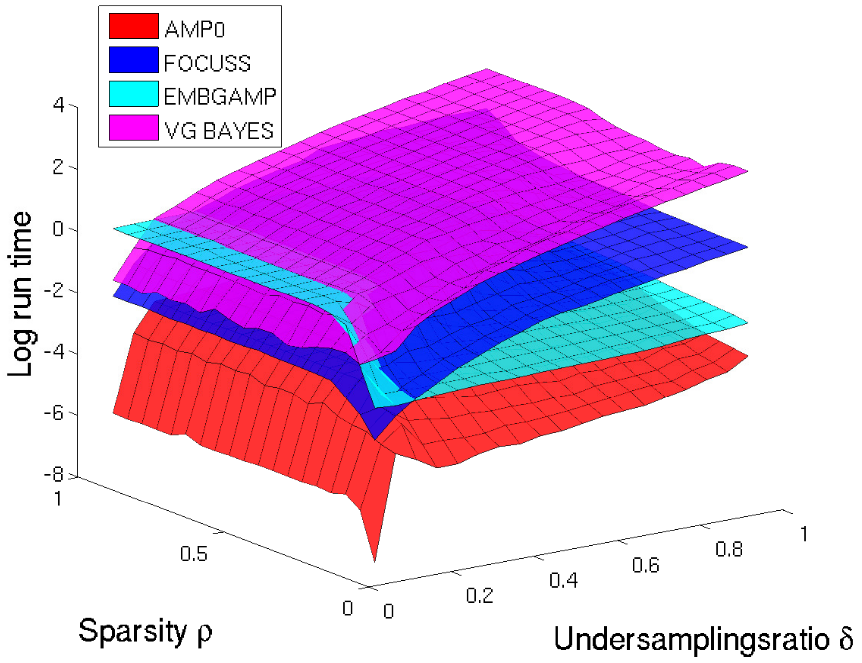
Figure 4.9 also clearly shows that the curve for EM-BG-AMP is well above all the other curves for all values of δ except for the point $\delta = 0.05$. Furthermore, for $\delta \geq 0.7$, the EM-BG-AMP method provides perfect reconstruction for all values of $\rho \in [0.05, 0.95]$. This is quite impressive, but perhaps not so useful in practice since many problems of interest are located in the opposite corner of the diagram, i.e. the lower left corner.



(a)

Figure 4.9: Estimated phase transition curves for the methods AMP0, EM-BG-AMP, VG Bayes, FOCUSS and GAMP2. The data are generated using $n = 500$, Gaussian I.I.D forward matrix, and the coefficients of all non-zero weights are fixed to 1. The values of δ and ρ are both sampled equidistant in the interval $[0.05; 0.95]$. The results are averaged over $R = 50$ runs.

We now consider the run times of the 4 different algorithms. The dominating operation in the AMP algorithm is forming the matrix-vector products involving $\boldsymbol{A}$ and $\boldsymbol{A}^T$, so AMP scales linearly w.r.t. both m and n, i.e. $\mathcal{O}(mn)$. In each iteration of both VG and FOCUSS, it is necessary to solve a linear system of equations, which scale as $\mathcal{O}(mn^2)$. It is therefore expected that AMP0 and EM-



(a)

Figure 4.10: The logarithm of the run time for AMP0, EM-BG-AMP, VG Bayes and FOCUSS. The data are generated as described in figure 4.9.

BG-AMP are faster than the two other methods. This is confirmed by the figure 4.10, which shows the natural logarithm of the CPU run time as function of both δ and ρ for the four methods. The natural logarithm is used because of the large difference in magnitudes. For almost all values of δ and ρ, the figure shows the following ordering: AMP0 $<$ EMBGAMP $<$ FOCUSS $<$ VG. Again, CPU run time is heavily affected by the specific implementation and hardware details and therefore one should be cautious to draw conclusions. Here we simply note that the observed run times are consistent with the expected results.

Before we move on to the analyzing the performance of the methods in a noisy setting, we briefly discuss the GAMP algorithm. Two variants of the GAMP algorithm, GAMP1 and GAMP2, have been introduced, where GAMP2 is a simplified version of GAMP1. It was argued that AMP0 can be seen as a special case of GAMP2 with specific choices of the scalar functions g_{in} and g_{out}. Figure 4.9 also shows the estimated phase transition curve for GAMP2, where the scalar functions have been chosen to match the AMP0 algorithm (see Appendix B.2). As seen on the figure, the phase transition curves for GAMP2 (black curve) and AMP0 (red curve) agree nicely for all values of δ.

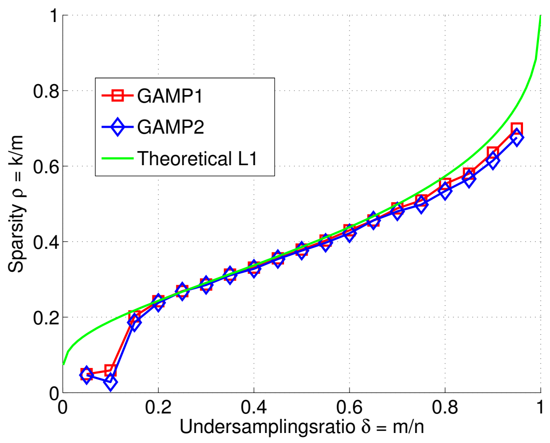
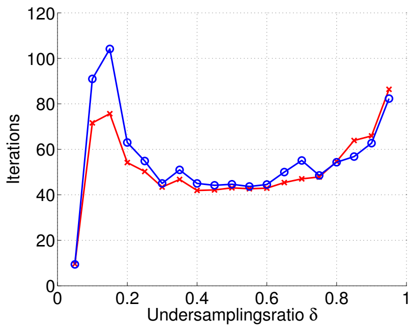
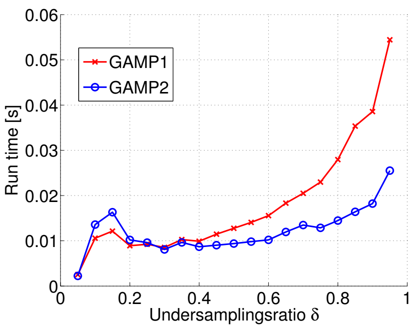


Figure 4.11: Estimated phase transition curves for GAMP1 and GAMP2, where the scalar function have been chosen as described in Appendix B.2. The data are generated the same way as described in 4.9.

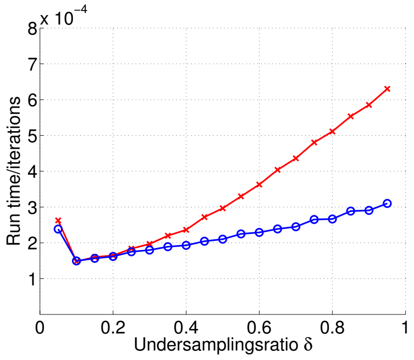
Moreover, figure 4.11(a) compares GAMP1 (red curve) and GAMP2 (blue curve) using these particular scalar functions. The data are generated in the same way as for the other phase transition curves described above. It is seen that the two curves are almost identical, but a careful look at the figure reveals that the red curve is slightly higher than the blue for most values of δ. This is also



(a) Iterations used



(b) Run time



(c) Run time/iteration

Figure 4.12: Number of iterations used and CPU run time for GAMP1 & GAMP2. The data are generated the same way as described in 4.9, except the sparsity is fixed to $\rho = 0.1$.

formally confirmed using a paired t-test, which yields a p-value of $p = 4.2710^{-4}$. Therefore, using a significance level of 1%, we reject the null hypothesis that the difference of the two curves has zero mean. However, as Rangan points out in [Ran10], this difference is expected to become insignificant when the problem size n increases.

Regarding the computational complexity, the dominating operation in both GAMP1 and GAMP2 is the matrix-vector products involving $\boldsymbol{A}$. GAMP1 requires forming four of such products, while GAMP2 only requires two of such products. Figure 4.12(a) shows the number of iterations used in GAMP1 and GAMP2 and figure 4.12(b) shows the CPU run time for the two methods. The data are generated the same way as described above, except that sparsity is fixed at $\rho = 0.1$. It is seen that the two methods require roughly the same number of iterations to converge. The peak around $\delta = 0.15$ is due to the transition between the unsolvable and solvable region. For small values of δ, the run times for the two methods are similar, but as δ increases the difference in run time between the two methods become large. Finally, figure (c) shows the run time per iteration for the two methods and here it is clearly seen that the proportionality constant for GAMP1 is approximately twice the size of the proportionality constant of GAMP2. The same tendency is exptected for the problem size n. This suggests that for problems with relative large δ, one should use GAMP2 rather than GAMP1 as expected.

Using the numerical experiments in this section, it has been shown that the AMP-based methods, AMP0 and EMBGAMP, outperform FOCUSS and VG both in terms of the phase transition curve and run time.

4.3 Noisy Problems

The purpose of this section is to examine the performance of the AMP methods, when the measurements are contaminated with noise. Both the AMPA algorithm and EM-BG-AMP algorithm are designed to handle noisy problems. However, the AMPA algorithm requires manually tuning of the regularization parameter λ, while the EMBG-AMP method automatically tunes the hyperparameter using an Expectation-Maximization scheme. In addition, the experiments in the noiseless case showed that the phase transition curve for EM-BG-AMP is superior to that of AMP0. Therefore, we will focus on the EM-BG-AMP method in this section.

Vila et al. have published a Matlab toolbox implementing the algorithm EM-

BG-AMP[3], which is based on the GAMP Matlab toolbox by Sundeep Rangan[4]. This implementation is used for all simulations involving the EM-BG-AMP methods.

In the noisy case, exact recovery is unlikely and therefore we cannot expect the coefficients of the estimated solution to converge to the true solution. Instead of labelling each run as successful or unsuccessfull as in the previous experiments, we will quantify the performance of an algorithm in terms of the Normalized Mean Square Error (NMSE) and the so-called F-measure from Information Retrieval [Rij79]. The NMSE measure is given by

$$NMSE = \frac{||\boldsymbol{x}_0 - \hat{\boldsymbol{x}}||_2^2}{||\boldsymbol{x}_0||_2^2} \tag{4.5}$$

The F-measure, also known as the F_1-score, corresponds to the harmonic mean of the *precision* and *recall*:

$$F = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \tag{4.6}$$

where precision is the fraction of true non-zero elements detected by the algorithm and recall is the fraction of true zero-elements detected by the algorithm. The F-measure will be used to quantify a given algorithms ability to estimate the correct location of the non-zero entries, i.e. the support of the solution. Note, $F = 1$ if and only if the support of the entire solution is correctly detected.

For the EM-BG-AMP method, we can estimate the support using the posterior probabilities of the support variables. That is, we will use the following MAP estimate:

$$\forall i \in [n]: \; \hat{s}_i = \arg\max_{s_i} p(s_i \big| \boldsymbol{Y}) \tag{4.7}$$

Sensitivity of Hyperparameters of BG-AMP

The BG-AMP method has four hyperparameters, where three of them governs the prior distribution on the entries in the solution $\boldsymbol{x}$ and the fourth is the noise variance. This aim of this experiment is to investigate how the specific value of a given hyperparameter influence the performance of the algorithm. That is, how sensitive is the algorithm to the specific values of the individual hyperparameters. It should be stressed that the EM update scheme is not used in this experiment.

[3] http://www2.ece.ohio-state.edu/~vilaj/EMGMAMP/EMGMAMP.html
[4] http://gampmatlab.wikia.com/wiki/Generalized_Approximate_Message_Passing

In the BG-AMP algorithm, the prior distribution on each of the elements x_i is assumed to have the form:

$$p(x_i|\boldsymbol{q}) = (1-\lambda)\,\delta(x_i) + \lambda\mathcal{N}\left(x_i|\zeta,\psi\right) \tag{4.8}$$

For this experiment the problem size is fixed to $n = 500$, undersamplingsratio $\delta = \frac{1}{4}$ and the sparsity is fixed to $\frac{1}{4}$. Then $R = 100$ problem instances are generated as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e}$. The true solutions $\boldsymbol{x}_0$ are sampled from the prior in eq. (4.8) with parameters $\lambda = \delta\rho = \frac{1}{16}$, $\zeta = 0$ and $\psi = 1$. The forward matrices $\boldsymbol{A}$ are I.I.D. Gaussian, where the columns have been scaled to unit ℓ_2-norm. The noise vectors $\boldsymbol{e}$ are generated from an multivariate isotropic Gaussian distribution $\mathcal{N}\left(0, 10^{-3}\boldsymbol{I}_m\right)$.

We will now investigate the influence of the sparsity parameter λ. To do this, we fix the other hyperparameter, i.e. ζ, ψ and σ^2, to the their respective true values and then sweep over λ in the range 10^{-10} to 1. Figure 4.13(a)-(b) show the NMSE and F-measure as a function of λ. The green vertical line indicates the true value of the hyperparameter. The thin dashed blue line show the result for one problem instance, while the solid blue line shows the result averaged over all $R = 100$ problem instances. Figures (c)-(d) show the same plots for the noise variance σ^2. Figure 4.14(a)-(b) show the same plots for the mean of the "slap" component ζ and finally figure 4.14(c)-(d) show the same plots for the variance of the "slap" component ψ. Note the different type of axis for each of the hyperparameter.

From the figures, it is clear that the optimal performance, both in terms of NMSE and F-measure, is obtained when the values of the hyperparameters are equal to their true values as expected. For the three hyperparameters related to the prior, i.e. λ, ζ and ψ, there is a relative large range of values, which results in optimal or near optimal performance. For instance, the BG-AMP method yields an F-measure close to 1, as long as λ is in the range $\left[10^{-7}, 0.5\right]$. At this point, it is important to emphasize that the other hyperparameters are fixed to their true values. But these plots still suggest that the algorithm is not that sensitive to the values of the hyperparameters of the prior. However, this is not as clear for the noise variance. The figures (c) and (d) show that the algorithm is a bit more sensitive to the value of the noise variance σ^2 than the remaining hyperparameters.

Figure 4.14(a) and (b) show that the performance of the algorithm is heavy fluctuating when ζ is far away from the true value. This is likely to be caused by numerical issues since the value of a Gaussian probability density function is extremely small when evaluated more than 5 standard deviations away from its mean. Figure 4.14(c) and (d) suggest that the exact value of the variance of the "slap" component is not important as long as the "slap" density is broad

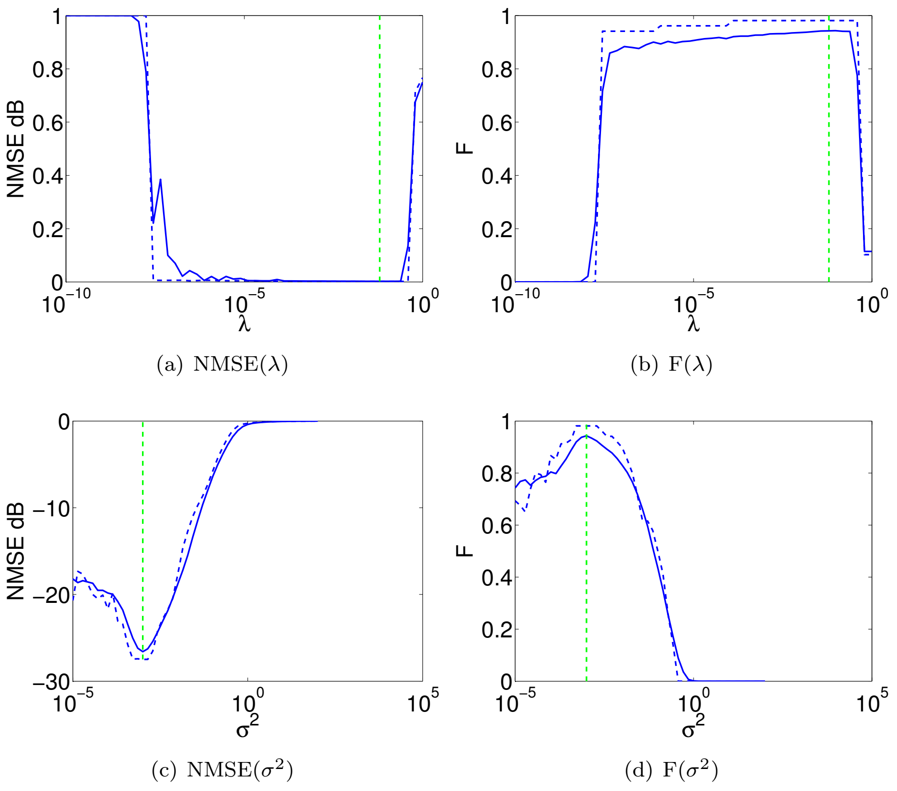
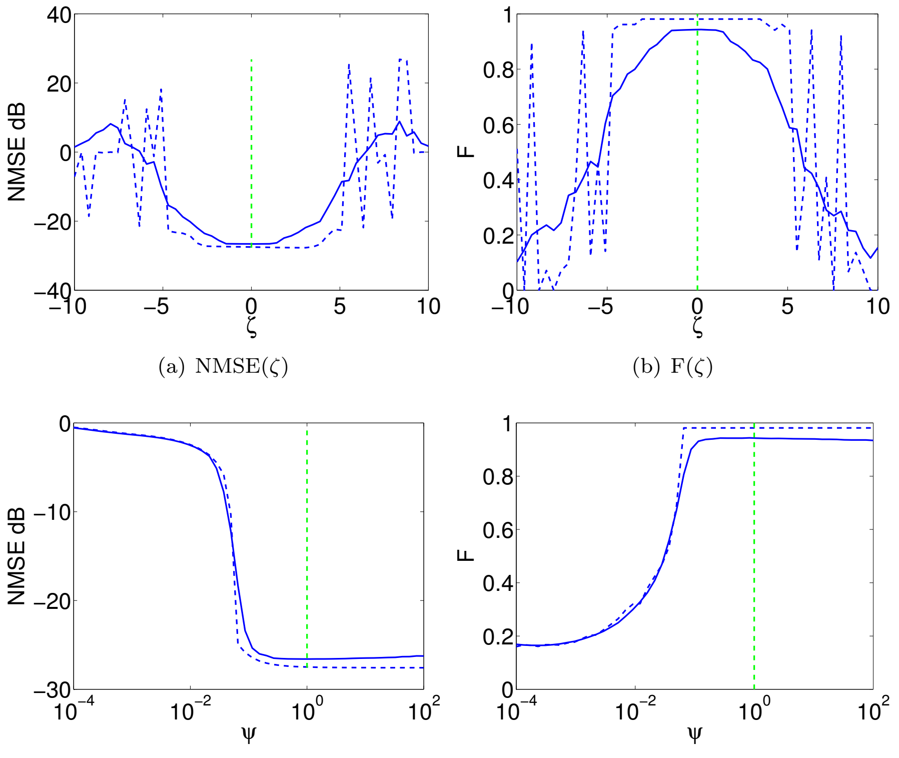


Figure 4.13: Sensitivity of λ and σ^2. The data is generated using $n = 500$, $\delta = \frac{1}{4}$, $\rho = \frac{1}{4}$, $\lambda = \frac{1}{16}$, $\zeta = 0$, $\psi = 1$ and $\sigma^2 = 10^{-3}$. (a)-(b) show NMSE and F as a function of λ, while the remaining parameters are fixed at the true values. The solid blue line is the average value over 100 runs, whereas the dashed line correspond to one instance and the dashed green line indicates the true value. (c)-(d) show similar plots for σ^2.



(a) NMSE(ζ) (b) F(ζ)

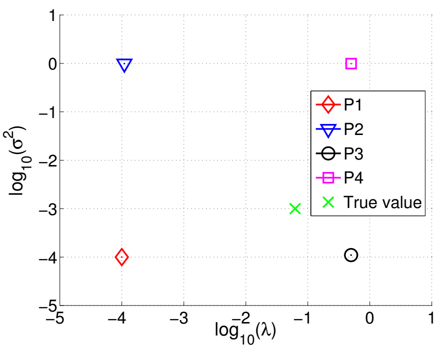
(c) NMSE(ψ) (d) F(ψ)

Figure 4.14: Sensitivity of ζ and ψ. The data is generated using $n = 500$, $\delta = \frac{1}{4}$, $\rho = \frac{1}{4}$, $\lambda = \frac{1}{16}$, $\zeta = 0$, $\psi = 1$ and $\sigma^2 = 10^{-3}$. (a)-(b) show NMSE and F as a function of ζ, while the remaining parameters are fixed at the true values. The solid blue line is the average value over 100 runs, whereas the dashed line correspond to one instance and the dashed green line indicates the true value. (c)-(d) show similar plots for ψ.

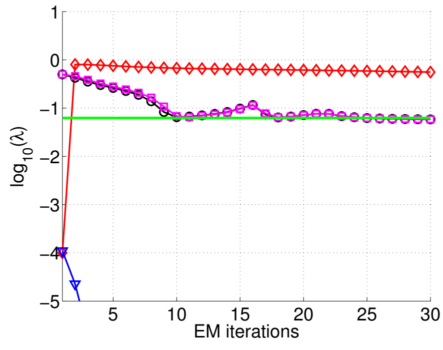
enough to cover the range of coefficients. Furthermore, it is also expected that as the value of ψ increases, the sensitivity to ζ decreases.

Learning the Hyperparameters using EM

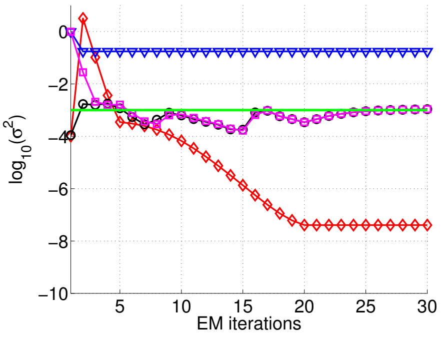
We will now consider the process of learning the hyperparameters using the EM update scheme. In particular, since the EM algorithm is only guaranteed to converge to a local maxima (or saddle point), we will investigate the algorithms sensitivity to the initial values. In order to do this, an experiment is set up with the same parameters as above, but now some of the hyperparameters are initialized to "wrong" values and then updated using the EM updates rules using the EM-BG-AMP algorithm.



(a)



(b)



(c)

Figure 4.15: Learning of the sparsity rate λ and the noise variance σ^2. The data set is generated from a model with the following parameters: $n = 2000$, $\delta = \frac{1}{4}$, $\rho = \frac{1}{4}$, $\lambda = \frac{1}{16}$, $\zeta = 0$, $\psi = 1$ and $\sigma^2 = 10^{-3}$. Except for λ and σ^2, the algorithm is provided with perfect prior knowledge. (a) shows the true values and the 4 initial points (b) shows the trajectories of λ^k for each of the initial points (c) shows a similar figure for $\sigma^2(k)$.

The mean and variance of the ”slap” component are initialized to the true values, while we consider different initializations for the pair (λ, σ^2). The four different points initializations (P1-P4) along with the true values (green cross) are shown in figure 4.15(a). The trajectory of the sparsity parameter λ is shown as a function of EM iterations in figure (b) and figure (c) show a similar plot for the noise variance σ^2. Each color corresponds to a different point of initialization and the green lines mark the true values. Observe that only the two initializations starting from point P3 and point P4 do actually converge.

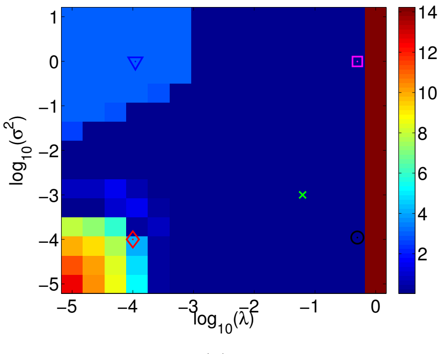
Consider first the initialization at point P1 (red), where both the sparsity parameter and the noise variance are too high. The EM-algorithm converges to a solution where λ has a relatively large value and the noise variance has a very small value as shown in figure (b). This can be interpreted as a case of overfitting, since most of the resulting coefficients are active, i.e. $\hat{x}_i \neq 0$.

The opposite situation happens when the algorithm is initialized at point P2 (blue). Here the sparsity parameter is too low and the noise variance is too high. As seen on figure (b) the sparsity parameter drops quickly during the first iterations and never ”returns”, while the noise variance converges at a relatively large value. This configuration correspond to a local maxima, where all coefficients are inactive, i.e. $\hat{x}_i = 0, \forall i \in [n]$, and the noise variance explains all the variation in the data.

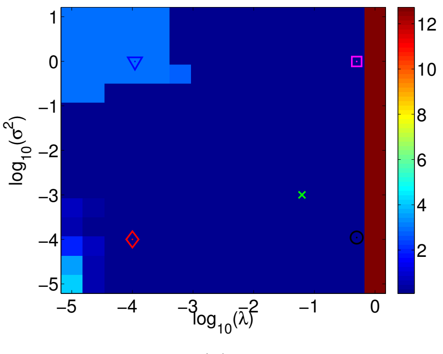
Finally, we see that when the algorithm is initialized at point P3 and P4, the estimated values of the hyperparameters converges to the true values. It is remarkable that the trajectories for the noise variance are identical for most of the iterations even if the initial values are very different.

Figure 4.15(b)-(c) show the trajectories of the two hyperparameters λ and σ^2 vs. number of EM iterations for the 4 different initial points shown in figure 4.15a. It is seen that the blue curve and the red curve seem to converge to poor local maximas. The initial point for the blue curve corresponds to very low sparsity rate and large noise variance and it converges to a solution, where all the variation in the data is explained by the noise. Next, we see that the red curve converges to a solution with a high sparsity rate and very low noise variance, which suggests overfitting. But the trajectories for the black and purple curves converge nicely to the true solution.

The fact that different initializations converge to the exact same point suggests that there is *bassin* of attraction in the (λ, σ^2)-space, in which the solutions converge to the same points. In order to investigate this further, a new experiment is set up. The space (λ, σ^2) is sampled on a regular grid with 15 points along each dimension. For every grid point $R = 10$ problems are generated and solved using EM-BG-AMP, where λ and σ^2 is initialized according to the current grid



(a)



(b)

Figure 4.16: Euclidean distance in $\mathbb{R}^2$ between the estimated parameters $\hat{\lambda}, \hat{\sigma}^2$ and corresponding true values. Same conditions as above, but averaged over $R = 10$ runs. (a) ζ, ψ are fixed at true (b) Initial values for $\zeta^0 = 0.5, \psi^0 = 10$ and updated using EM

point. The mean and variance of the "slap" component is still initialized to the true values. As a measure of performance, we will use the euclidean distance between the estimated hyperparameters $(\hat{\lambda}, \hat{\sigma^2})$ and the true hyperparameters (λ, σ^2). The result is shown in figure 4.16(a). The entire experiment is repeated one more time, except now we initialize (ζ, ψ) as $\zeta = 0.5$ and $\psi = 10$, both of which are "incorrect" values.

As expected, both figure (a) and (b) show a large bassin of attraction, where the estimated hyperparameters are close to the true values. Additionally, figure (b) verifies that initialization of (ζ, ψ) is not crucial for the result. At least not when the initial value of ψ is large.

Performance of EM-BG-AMP vs. LASSO vs. VG

This subsection describes an experiment, whose purpose is to compare the performance of EM-BG-AMP to other methods. In particular, we will compare EM-BG-AMP to the LASSO [Tib96] and the Variational Garrote [KG12]. The LASSO is widely known and used as reference for many experiments and is therefore relevant in a comparison. The Variational Garrote is a new promising method, which have been shown to outperform the LASSO on multiple occasions [KG12, HSH13]. For the LASSO, we will use the Least Angle Regression-variant (LARS) [EHJT], which is implemented by Karl Sjostrans in the Matlab toolbox [Sjö05]. The LARS algorithm computes the entire regularization path of LASSO in an efficient manner. For the Variational Garrote (VG), we will use the implementation in the FieldTrip Matlab toolbox [OFMS11]. In order to tune the regularization parameter γ for VG, 15% of the samples are used for hold-out cross validation[5] [TSK06]. For EM-BG-AMP, the maximum number of EM iterations is fixed to 60, while the maximum number of AMP iterations is fixed to 25.

The experiment is conducted as follows. The data are generated from a model of the form $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e}$. The problem size is again fixed to $n = 500$ and the forward matrix is I.I.D Gaussian, where the columns have been scaled to unit ℓ_2-norm. The true solutions $\boldsymbol{x}_0$ are generated such that they have $k = \rho \cdot m$ (or closest integer) non-zero entries[6], and the coefficients of the non-zero entries are sampled from a zero-mean Gaussian distribution with unit variance. The noise is sampled from an isotropic multivariate Gaussian distribution, $\boldsymbol{e} \sim \mathcal{N}\left(0, \sigma^2 \boldsymbol{I}_m\right)$. The noise variance σ^2 is scaled to fit a desired signal-to-noise ratio (SNR) using:

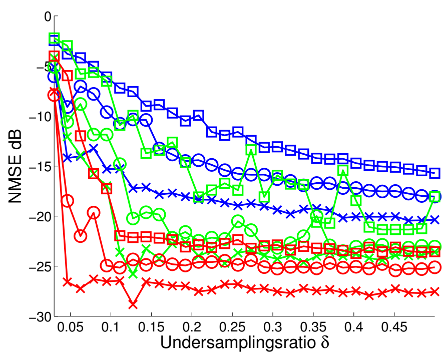
$$\text{SNR}_{dB} = 10 \log_{10} \frac{\mathbb{V}\left[\boldsymbol{A}\boldsymbol{x}_0\right]}{\mathbb{V}\left[\boldsymbol{e}\right]} \tag{4.9}$$

The undersamplingsratio δ is sampled 30 times with equal spacing in the interval $[0.025, 0.5]$, the sparsity ρ is chosen to be $\rho \in \{0.10, 0.15, 0.20\}$ and the SNR is chosen to $SNR \in \{10dB, 15dB, 20dB\}$. For each combination of (δ, ρ, SNR), we generate $R = 100$ problem instances and solve them using EM-BG-AMP, LASSO and VG. The hyperparameters of EM-BG-AMP are initialized as described in Algorithm 5.

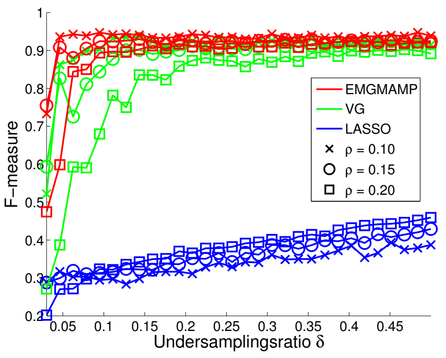
The results are shown in figure 4.17(a)-(f). The red curves correspond to EM-BG-AMP, the green curves are VG and the blue curves are LASSO. The markers on each line indicate the current sparsity, i.e. a cross is $\rho = 0.10$, a circle is $\rho = 0.15$ and a square is $\rho = 0.20$. In each of the six figures and for all three methods, it is seen that the performance improves when the sparsity parameter ρ decreases and when the signal-to-noise ratio increases as expected. Furthermore, we also see that in all six figures EM-BG-AMP outperforms VG and VG outperforms the LASSO. Inspecting figure (b), it is seen that for $\delta > 0.2$, the F-measure for both VG and EM-BG-AMP are approximate 0.9 for all three degrees of sparsity. This implies that both methods recovers most of the correct non-zero entries in $\boldsymbol{x}_0$. But by inspecting the range where $\delta < 0.2$, it is seen that the degree of sparsity has huge impact on the performance of VG, while it has much less

[5]Here we tune the regularization parameter γ using cross validation rather than learning it from the data, since personal experience has shown that the cross validation is more robust to noise.

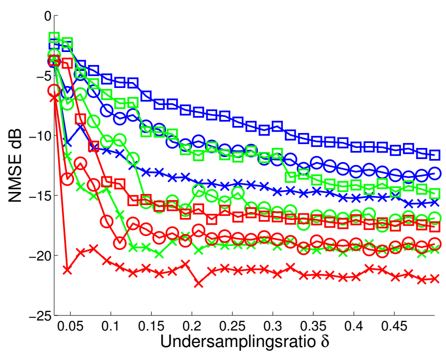
[6]Ideally, the number of non-zero components should be sampled from a Binomial prior distribution with parameter $\lambda = \rho\delta$, but for small values of δ this approach results in many realization of $\boldsymbol{x}_0$, where all entries are inactive.



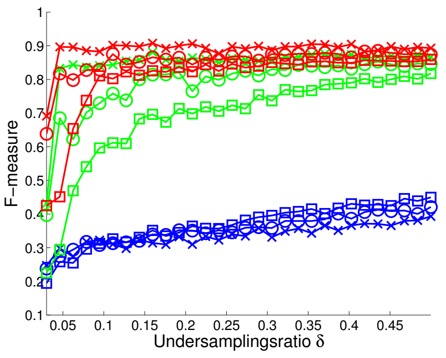
(a) NMSE for $SNR = 20dB$



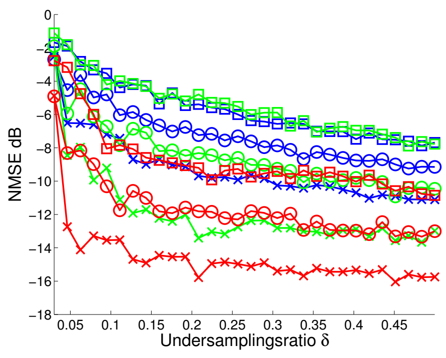
(b) F-measure for $SNR = 20dB$



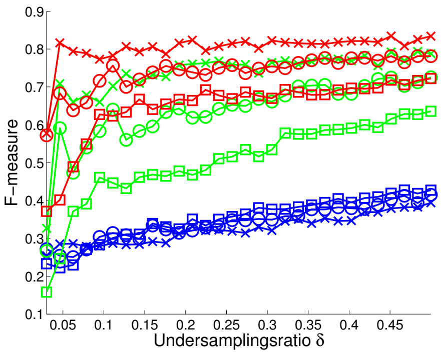
(c) NMSE for $SNR = 15dB$



(d) F-measure for $SNR = 15dB$

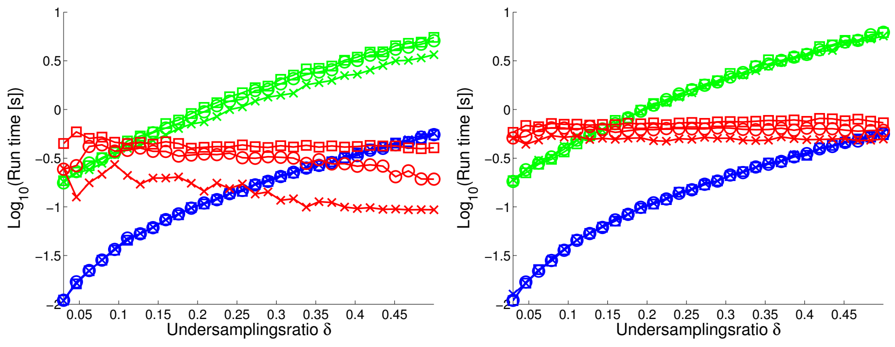


(e) NMSE for $SNR = 10dB$



(f) F-measure for $SNR = 10dB$

Figure 4.17: Comparison of EM-BG-AMP, LASSO and VG for different undersamplingsratios, sparsity levels and signal to noise ratios. Data are generated using a I.I.D. Gaussian forward matrix $\boldsymbol{A}$, the solutions $\boldsymbol{x}_0$ are generated such that they have $k = \rho \cdot \delta \cdot n$ non-zero entries and the coefficients of the non-zero elements are samples from a $\mathcal{N}(0, 1)$ distribution. The results are averaged over $R = 100$ runs.



(a) Run time for $SNR = 20dB$ (b) Run time for $SNR = 10dB$

Figure 4.18: Comparison of run times for the EM-BG-AMP, LASSO and VG. The details of the experiment are the same as for figure 4.17

.

effect on the performance of EM-BG-AMP. The is an important detail, since the performance on problem with a very high degree over undersampling, i.e. very small δ, is crucial for applications like EEG imaging.

The CPU run times for this experiment is shown in figures 4.18(a)-(b) for SNR$= 10dB$ and SNR$= 20dB$. The plot shows the logarithm of the run times measured in seconds. It is seen that for small values of δ, the LARS-algorithm is by far the fastest method, although it performance poorly in terms of NMSE and F-measure. It also appears that for small values of δ, the EM-BG-AMP method is the slowest of three methods, but it clearly has the best scaling in terms of δ. For $\delta > 0.2$ EM-BG-AMP outperfroms VG in terms of run time. Also note that the degree of sparsity ρ has greater influence on the run time for EM-BG-AMP than for the two other algorithm.

The prior distribution on $\boldsymbol{x}$ in VG and EM-BG-AMP are very similar. Recall, that each entry of the solution x_i can be decomposed into a support variable $s_i \in \{0, 1\}$ and a coefficient variable $\theta_i \in \mathbb{R}$, where $x_i = s_i \cdot \theta_i$. The VG model assumes a Bernoulli prior distribution on each s_i and an improper constant distribution on each coefficient θ_i. The Bernoulli distribution is parametrized using a common regularization parameter γ, which is tuned using hold-out cross validation. The posterior distribution $p(\boldsymbol{x}|\boldsymbol{y})$ is then obtained using a mean-field variational approximation. The EM-BG-AMP model also imposes a Bernoulli distribution on each $\boldsymbol{s}_i$, while a Gaussian distribution is imposed on the coefficients θ_i. But here the desired posterior $p(\boldsymbol{x}|\boldsymbol{y})$ is obtained using the GAMP approximation and the hyperparameters are estimated from the data using EM scheme. Given the similarities of the two models, it is interesting to speculate what actual causes the difference in performance.

First, the effective number of measurements available for VG is $\hat{m} = 0.85m$, since 15% of the samples are used for cross validation. This implies that

$$\hat{\delta} = \frac{\hat{m}}{n} = \frac{0.85m}{n} = 0.85\delta \tag{4.10}$$

$$\hat{\rho} = \frac{k}{\hat{m}} = \frac{k}{0.85m} = \frac{1}{0.85}\rho \approx 1.17\rho \tag{4.11}$$

We can interpret this as the effective undersamplingsratio is decreased, while the effective sparsity parameter is increased and thus making the underlying inverse problem more difficult to solve (see the estimated phase transition curves in section 4.2). Second, the VG model assumes an uninformative (improper) constant distribution on the coefficients, while the EM-BG-AMP models assumes a Gaussian distribution on the coefficients. From a philosophical and theoretical point of view, the uninformative constant distribution is appealing since no prior information on the coefficients is available, but perhaps the Gaussian distribution in the EM-BG-AMP model also has a regularization effect as in Ridge regression [McD09]. Third, two different types of approximations are used in order to obtain the desired posterior distribution. It might be that the AMP approximation has better inherent properties than the mean-field approximation used in the VG method for this particular type of problem.

4.4 Multiple Measurement Vector Problems with Static Support

The goal of this section is to examine the properties of the AMP-MMV algorithm. Justin ziniel et al. [ZS13b] have implemented the EM-AMP-MMV model in a Matlab toolbox[7], which we will utilize for all experiments involving the AMP-MMV model.

Three different experiments are carried out using the EM-AMP-MMV model. The first experiment simply illustrates the effect of having multiple measurement vectors available, while the second experiment investigates the influence of the number of EM iterations for EM-AMP-MMV. Finally, in the third experiment the performance of EM-AMP-MMV is compared to TM-SBL [ZR13], M-SBL [WR07] and M-FOCUSS [CREKD05] in a systematic fashion.

All three methods (TM-SBL, M-SBL, M-FOCUSS) are available in the Matlab

[7] http://www2.ece.ohio-state.edu/~zinielj/mmv/

toolbox implemented by Zhilin Zhang[8]. To facilitate comparison, the TM-SBL, M-SBL and M-FOCUSS are configured the same way as in [ZS13b]. For the TM-SBL methods, the noise parameter is configured according to the current SNR:

$$\texttt{noise} = \begin{cases} \texttt{small} & \text{if } 22dB < SNR \\ \texttt{mild} & \text{if } 6dB < SNR < 22dB \\ \texttt{large} & \text{if } SNR < 6dB \end{cases}$$

For the M-SBL method, the regularization parameter is estimated from the data and the initial choice of the regularization parameter is fixed to $\lambda = 10^{-3}$. The pruning threshold is set to 10^{-2}, the convergence tolerance is set to 10^{-8} and the maximum allowed number of iterations is 2000. For the M-FOCUSS method, we simplify fix the regularization parameter to the true noise variance, which is an optimal or near optimal value. This configuration is identical to the configuration used in [ZS13b] and is used throughout all the following experiments.

For the AMP-MMV method, we can use the posterior probabilities of $s_i = 1$, to make a MAP estimate of the support, i.e.

$$\forall i \in [n] : \; \hat{s}_i = \arg\max_{s_i} p(s_i \big| \boldsymbol{Y}) \tag{4.12}$$

But such posterior quantities are not available for the other three methods. Instead, we will use a rather optimistic estimate of the support. Let k denote the number of non-zero rows in the true solution matrix $\boldsymbol{X}$. We will then compute the ℓ_2-norm of each row of the estimated solution matrix $\hat{\boldsymbol{X}}$ and use the location of k rows with the largest ℓ_2-norms as the support estimate. In practice, the true value of k is unknown, so this estimate is indeed optimistic. However, this type of estimate is used in [ZS13b, ZR13] and therefore it is also adopted here to facilitate comparison.

In the following experiments, we consider the noisy MMV problem of the form:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{E} \tag{4.13}$$

where $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^1 & \boldsymbol{x}^2 & \dots & \boldsymbol{x}^T \end{bmatrix}$ is the true solution. To quantify the performance of the algorithms, the NMSE measure is extended to the Time Normalized Mean Square Error (TNMSE), which is given by:

$$TNMSE = \frac{1}{T}\sum_{t=1}^{T} \frac{||\boldsymbol{x}^t - \hat{\boldsymbol{x}}^t||_2^2}{||\boldsymbol{x}^t||_2^2}, \tag{4.14}$$

[8]http://dsp.ucsd.edu/~zhilin/TMSBL.html

where $\boldsymbol{x}^t$ are the true solutions and $\hat{\boldsymbol{x}}^t$ are the estimated solutions. To quantify the support recovery capabilities of the algorithms, the F-measure will still be used, but now the precision and recall are computed for the entire matrix $\hat{\boldsymbol{X}}$ instead of the vectors $\hat{\boldsymbol{x}}^t$.

The Effect of Multiple Measurement Vectors

The first experiment is designed to illustrate the benefit of having multiple measurement vectors available. The experiment is conducted as follows. The problem size is fixed to $n = 1000$ and the sparsity is fixed to $\rho = 0.3$. The undersamplingsratio δ is then sampled 10 times in the interval $[0.01, 0.15]$ with equal spacing. Then using an I.I.D. Gaussian forward matrix $\boldsymbol{A}$, where the columns have been scaled to unit ℓ_2-norm, we generate measurements using

$$\boldsymbol{y}^t = \boldsymbol{A}\boldsymbol{x}^t + \boldsymbol{e}^t \tag{4.15}$$

where the true solutions $\boldsymbol{x}^t$ are sampled from the prior of AMP-MMV model with the following hyperparameters

$$\zeta = 0 \quad , \psi = 1, \quad \alpha = 0.9, \quad \lambda = \rho\delta, \quad \sigma_c^2 = 1 \tag{4.16}$$

Recall that the prior model of AMP-MMV is:

$$\begin{aligned} p(s_i = 1) &= \lambda \\ p(\theta_i^t) &= \mathcal{N}\left(\theta_i^t | \zeta, \sigma_c^2\right) \end{aligned}$$

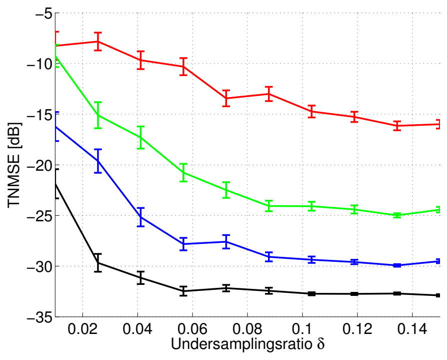
with the temporal evolution of the coefficients

$$\theta^t = (1 - \alpha)\left(\theta^{t-1} - \zeta\right) + \alpha w^t + \zeta$$

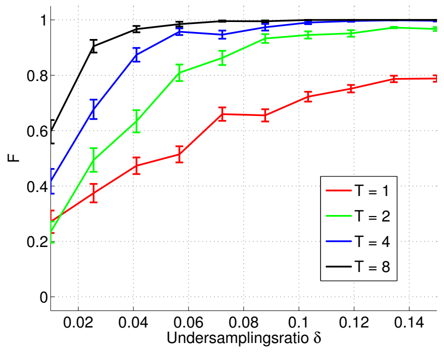
where the variance of the driving noise $w^t \sim \mathcal{N}(0, \rho)$ is determined by relation $\sigma_c^2 = \frac{\alpha\rho}{2-\alpha}$. The noise variance σ^2 is scaled such that the signal to noise ratio for each *time frame* is SNR $= 20dB$. Then for each value of T in $T \in \{1, 2, 4, 8\}$, we generate $R = 100$ problems . When $T = 1$, the BG-AMP method is used and for $T > 1$, the AMP-MMV method is used.

Both methods are given perfect prior information. The number of AMP iterations is fixed to 25 and the number of EM iterations is fixed to 15 for both methods. Figure 4.19 shows a plot of TNMSE and F-measure as a function of undersamplingsratio δ and figure 4.20 shows the same, but as a function of the signal-to-noise ratio. In the latter case the undersamplingsratio is fixed to $\delta = 0.2$.

Inspecting the figures clearly reveals that for a fixed value of δ or SNR, the recovery performance increases when more measurement vectors are available. For

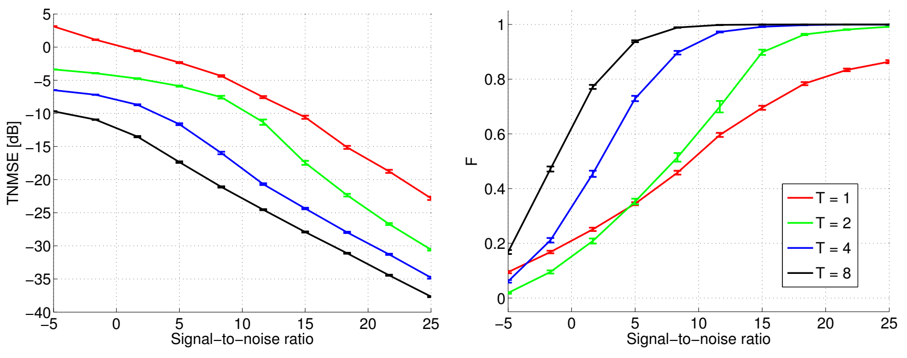


(a) TNMSE



(b) F measure

Figure 4.19: Performance measures, TMNSE and F-measure, as a function of undersamplingsratio δ for data sampled from the model: $n = 1000, \rho = 0.3$ and σ^2 is scaled s.t. the SNR for each *timeslice* is approximately $20dB$. The true parameters for the prior model is $\zeta = 0, \psi = 1$ and $\alpha = 0.9$. or $T = 1$, the EM-BG-AMP routine is used and for $T > 1$ the AMP-MMV routine is used. Both methods have perfect prior information. For both methods, a maximum of 15 *EM iterations* and 25 *AMP iterations* are allowed. The results are averaged over $R = 100$ runs.



(a) TNMSE (b) F measure

Figure 4.20: Performance measures, TMNSE and F-measure, as a function of SNR for data sampled from the model: $n = 1000, \rho = 0.3$ and $\delta = 0.2$. The true parameters for the prior model is $\zeta = 0, \psi = 1$ and $\alpha = 0.9$. or $T = 1$, the EM-BG-AMP routine is used and for $T > 1$ the AMP-MMV routine is used. Both methods have perfect prior information. For both methods, a maximum of 15 *EM iterations* and 25 *AMP iterations* are allowed. The results are averaged over $R = 100$ runs.

instance, figure 4.19(b) shows that for $\delta = 0.06$, the F-measure is approximately 0.5 for $T = 1$, but for $T = 8$ the F-measure is almost 1. A similar phenomena is observed for the TNMSE on figure (a). Consider now figure 4.20(a). Here it is seen that for $T = 1$, we need a signal-to-noise ratio of approximately 20dB to obtain an F-measure of approximately 0.8, while for $T = 8$, we can settle for SNR$\approx 3dB$ to achieve the same F-measure.

This simple experiment illustrates that problems, that are impossible to solve in the case $T = 1$ due to a high degree of undersampling, become solvable as T increases. Informally, as T grows we can use more and more information to estimate the support of the solution and thus the problems become easier to solve.

Number of Smoothing Iterations

The next experiment is set up to investigate the effect of the number of EM iterations in the EM-AMP-MMV model. In this experiment the data are generated using the same procedure as described for the previous experiment. The parameters of the experiment are also the same except, here δ is in range $[0.02, 0.4]$ and the number of measurement vectors is fixed to $T = 4$. For each value of δ, we generate $R = 100$ problems and solve them using EM-AMP-MMV with a different number of EM iterations. The hyperparameters are initialized as follows:

$$\lambda = \rho_{SE}(\delta) \cdot \delta, \quad \zeta = 0, \quad \sigma_c^2 = 1.5, \quad \sigma_e^2 = 0.5, \quad \alpha = 0.5, \tag{4.17}$$

where $\rho_{SE}(\delta)$ is the theoretical phase transition curve for ℓ_1 minimization (see literature review 1.2). In order to have a frame of reference, the result for the M-SBL method is also included in the graph. The results are shown in figure 4.21.

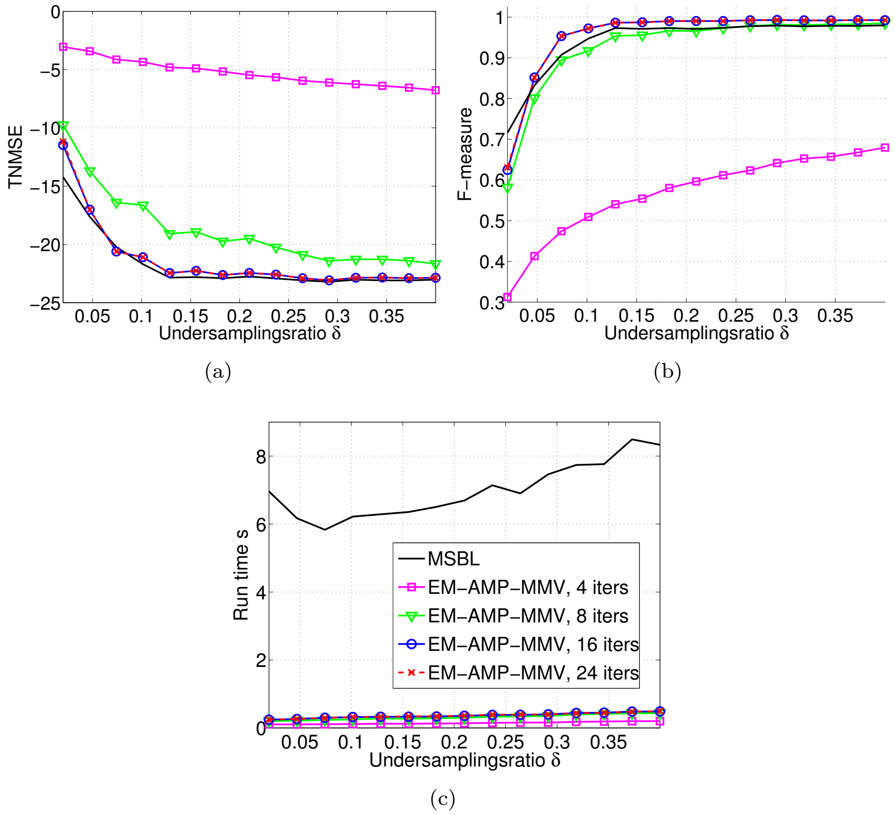


Figure 4.21: The effect of the number of smoothing iterations with MSBL as reference. The data are generated using the following parameters: $n = 1000$, $\rho = 0.3$, SNR = 20dB, $zeta = 0$, $\psi = 1$, $\alpha = 0.9$. The results are averaged over $R = 100$ runs.

It is clearly seen that 4 EM iterations is far from sufficient for the estimated hyperparameters to converge to reasonable values. On the other hand, no improvement is achieved when the number of EM iterations is increased from 16 to 24. Therefore, in the following experiments, we will set the maximum number of allowed EM iteration to at least 15.

Performance vs. Other Algorithms

In this experiment, the EM-AMP-MMV algorithm is compared to the MSBL, TM-SBL and M-FOCUSS algorithms in a systematic fashion. The performance will be measured using TNMSE and F-measure as a function of undersamplingsratio δ, sparsity ρ, signal-to-noise ratio SNR and number of measurement vectors T.

In all simulations, the data are generated using an I.I.D. Gaussian forward matrix, where the columns have been scaled to unit ℓ_2-norm and $n = 1000$, $T = 4$, $\delta = 0.2$, and $\rho = 0.3$. This is the base set up for all the experiments is this subsection. If not specified otherwise, these parameters are used in all the simulations. For each $t = 1, .., T$, the noise variance is scaled to provide a fixed SNR of 20dB. The true solutions $\boldsymbol{X}$ are sampled from the prior of the AMP-MMV model with the following hyperparameters:

$$\lambda = \rho\delta, \quad \zeta = 0 \qquad \sigma_c^2 = 1 \qquad \alpha = 0.1 \tag{4.18}$$

The EM-AMP-MMV algorithm is configured to use 25 AMP-iterations, 15 EM/smoothing iterations and with the following initialization of the parameters:

$$\lambda(0) = \rho_{se}(\delta) \cdot \delta, \quad \zeta(0) = 0, \quad \sigma_c^2(0) = 1.5, \quad \sigma_e^2(0) = 0.5, \quad \alpha(0) = 0.9 \tag{4.19}$$

The reasoning behind this particular choice of initial values are as follows. The initial estimate should be reasonable such that the EM-scheme has a change to converge to a decent maxima, but the initial estimates should not be too close to the true values, since that would give the EM-AMP-MMV algorithm an unfair "edge" compared to the other algorithms. All results are averaged over $R = 100$ runs.

The figures 4.22(a)-(c) show the TNMSE, F-measure and CPU run times for the four methods as a function of the undersamplingsratio δ, respectively. As expected, the performance of all four methods degrade as the problem becomes more and more undersampled. Figure (a) and (b) show that the M-FOCUSS method has the highest error and lowest F-measure for most values of δ.

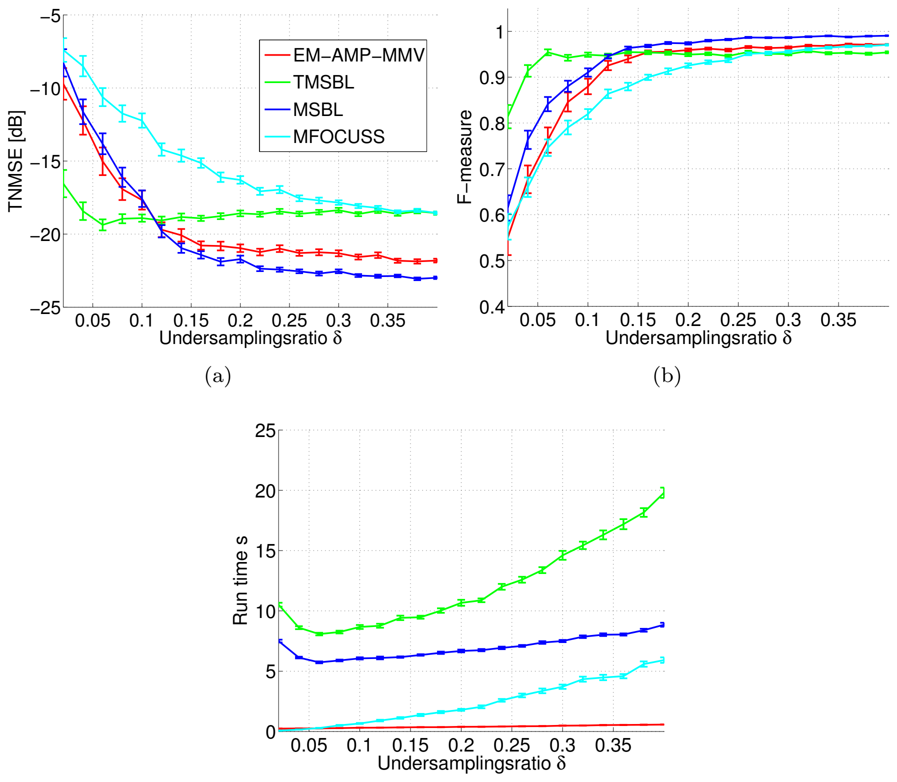


Figure 4.22: Performance vs. undersamplingsratio. The data are generated using an I.I.D. Gaussian forward matrix and the following parameters: $n = 1000$, $\rho = 0.3$, $T = 4$, SNR $= 20$dB, $\zeta = 0$, $\psi = 1$, $\alpha = 0.1$. The resuls are averaged over $R = 100$ runs.

Moreover, the figures show that for problems with a high degree of undersampling, i.e. $\delta < 0.1$, the TM-SBL method outperforms the other methods both in terms of TNMSE and F-measure. But for $\delta > 0.1$, the M-SBL outperform the other methods. This suggests that the TM-SBL has a model bias, which improves the solutions for small amounts of data, but restricts the model as more data become available. For both TNMSE and F-measure, the difference in performance between M-SBL and EM-AMP-MMV is rather small.

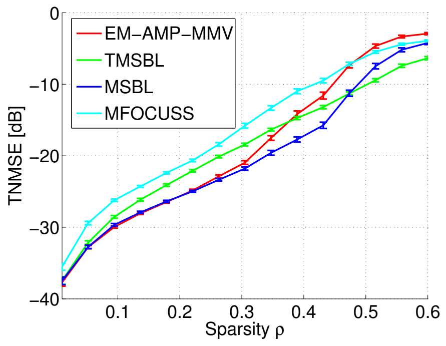
Figure (c) reveals the problem with many SBL-based methods. In terms of run time, the EM-AMP-MMV clearly outperforms the other methods. The computational complexity for each iteration of the EM-AMP-MMV method scales linearly in all problem dimensions, i.e. $\mathcal{O}(mnT)$. In the underdetermined setting, i.e. $m < n$, the computational complexity of M-FOCUSS and M-SBL are both $\mathcal{O}(nm^2)$ for each iteration [WR07], while TM-SBL has the extra computational load of estimating the correlation structure of the sources [ZR13]. The ordering of the analytical computational complexities of the methods are therefore consistent with the observed run times in figure (c).

Figures 4.23(a)-(c) show the TNMSE, F-measure and CPU run times for the 4 methods as a function of the sparsity ρ. As a sanity check, we see that the performance of all the methods decrease as ρ increases. Figure (a) shows that for $\rho < 0.1$, the TNMSE error of the four methods are very similar. But for $0.1 < \rho < 0.3$, the M-SBL and EM-AMP-MMV outperform the two other methods. For $0.3 < \rho$ the two SBL-based methods achieves the lowest errors.

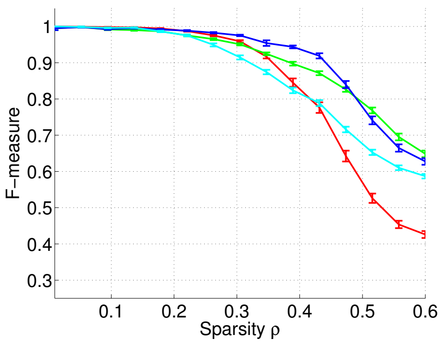
By inspecting figure (b), it is seen that for sufficiently sparse problems, i.e. small ρ, all four methods are able to recover the true support of the solution. But as the sparsity parameter ρ becomes greater than 0.2, the performance of all methods degrade significantly. Note, that for $\rho > 0.3.5$, the F-measure of EM-AMP-MMV drops rapidly compared to the other methods. This is likely to explained to the optimistic estimate of the support for the other methods. Figure 4.23(c) basically tells the same story as in figure 4.22(c).

Figures 4.24(a)-(c) show the TNMSE, F-measure and CPU run times for the four methods as a function of the number of measurement vectors T. For the MMV problem, there are different ways of obtaining more data. One can either increase the number of measurement per measurement vector, i.e. increase m, or one can increase the number of measurement vectors, i.e. increase T. With this comment in mind, we now compare the results in figure 4.24 to the results in figure 4.22. We see that increasing T essentially has the same effect as increasing m, which is of course one of the main benefits of the MMV formulation.

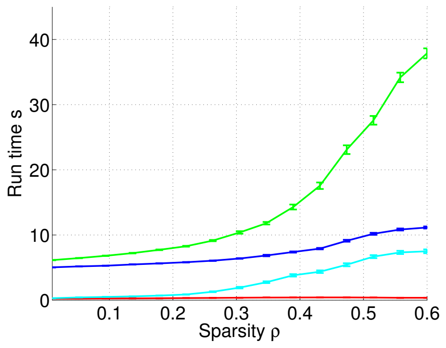
Consider this in the context of EEG imaging (see appendix D for more details). There are multiple reasons for keeping the number of electrodes m as low as



(a)

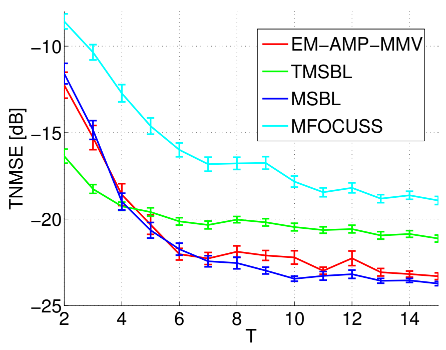


(b)

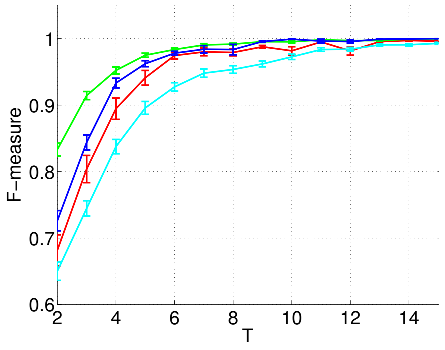


(c)

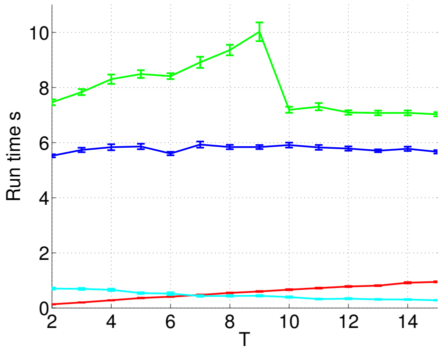
Figure 4.23: Performance vs. sparsity. The data are generated using an I.I.D. Gaussian forward matrix and the following parameters: $n = 1000$, $\delta = 0.2$, $T = 4$, SNR $= 20$dB, $\zeta = 0$, $\psi = 1$, $\alpha = 0.1$. The results are averaged over $R = 100$ runs.



(a)



(b)



(c)

Figure 4.24: Performance vs. T. The data are generated using an I.I.D. Gaussian forward matrix and the following parameters: $n = 1000$, $\delta = 0.2$, $\rho = 0.3$, SNR = 20dB, $\zeta = 0$, $\psi = 1$, $\alpha = 0.1$. The results are averaged over $R = 100$ runs.

possible, e.g. the surface area of the human scalp is rather small, it is time consuming to attach the electrodes properly etc. But once the electrodes are attached, multiple measurement vectors can easily be obtained without extra cost. However, the underlying dynamic nature of the brain enforces a relatively low bound on T. That is, if T becomes too large, the common sparsity assumption is no longer valid. But as seen in the figure, increasing T by two has tremendous effect on the recovery performance for all of the methods.

From figure 4.24(c) we also note that the M-FOCUSS methods actually becomes faster than EM-AMP-MMV for sufficiently large T. But keep in mind that in this experiment, we simply use the optimal regularization parameter for M-FOCUSS. That is, if the regularization parameter was to be tuned using cross-validation or similar, it would increase the run time significantly. Notice also the discontinuity in the curve for the TM-SBL method.

Figures 4.25(a)-(c) show the TNMSE, F-measure and CPU run times for the four methods as a function of the signal-to-noise ratio. At first glance figure (a) seems to suggest that EM-AMP-MMV is much better than the other methods in terms of TNMSE. But this is not the case since TNMSE$= 0dB$ amounts a relative error of 1. Thus, the EM-AMP-MMV simply returns $x_i^t = 0$ for all i and t, which is indeed desired compared to returning a useless non-zero solution. The evolution of the TNMSE and F-measure for all four methods are comparable and for this particular problem configuration, all four methods are virtually useless when the signal-to-noise ratio are below approximately 10dB.

Finally, figures 4.26(a)-(c) show the TNMSE, F-measure and CPU run times for the four methods as a function of the *inverse* correlation parameter α. It is seen that when $\approx\approx 1$, i.e. no correlation, all four methods are capable of reconstruction the underlying solution. But as α decrease, i.e. the temporal correlation increase, the performance of all four methods degrade, both in terms of TNMSE and F-measure. However, we note that EM-AMP-MMV algorithm is performing significantly better than the remaining three models for $\alpha < 10^{-2}$.

We will now summarize on the results in figures 4.22, 4.23, 4.24, 4.25 and 4.26. In the big picture, the performance in terms of TNMSE and F-measure for M-SBL, TM-SBL and EM-AMP-MMV are roughly the same, but the EM-AMP-MMV method are much faster than the other three methods. TM-SBL or M-SBL are able to achieve a slightly better TNMSE error, but it is at the cost of a huge increase in run time. We also note that the results are consistent with the results obtained in [ZS13b].

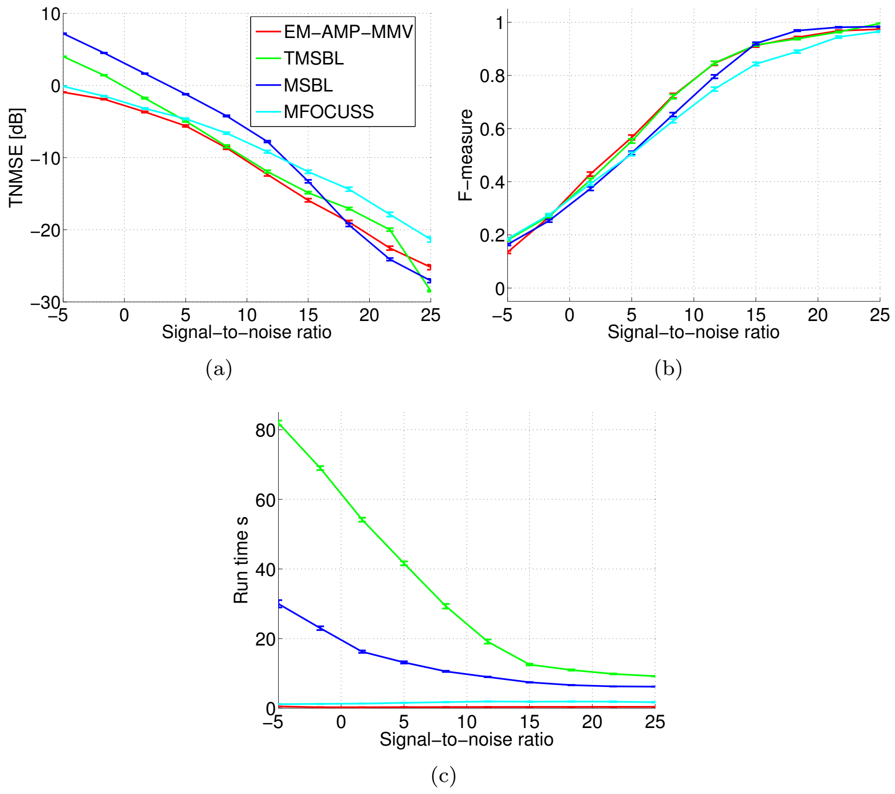
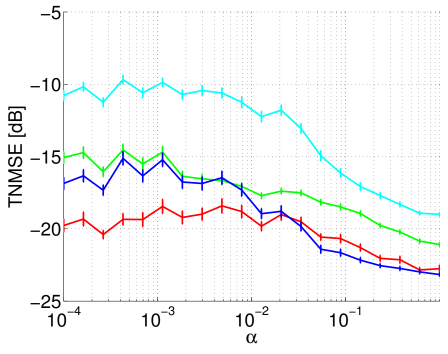
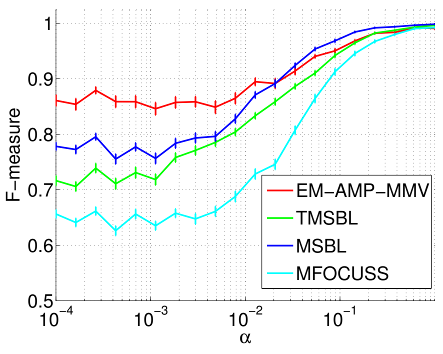


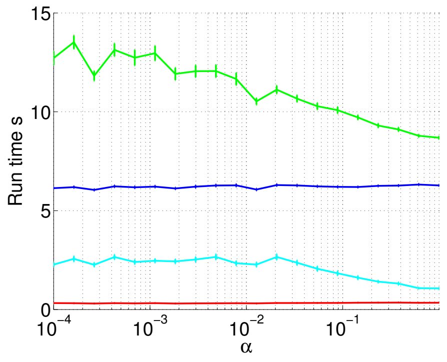
Figure 4.25: Performance vs. signal-to-noise ratio. The data are generated using an I.I.D. Gaussian forward matrix and the following parameters: $n = 1000$, $\delta = 0.2$, $\rho = 0.3$, $T = 4$, $\zeta = 0$, $\psi = 1$, $\alpha = 0.1$. The results are averaged over $R = 100$ runs.



(a)



(b)



(c)

Figure 4.26: Performance vs. *inverse* correlation parameter α. The data are generated using an I.I.D. Gaussian forward matrix and the following parameters: $n = 1000$, $\delta = 0.2$, $\rho = 0.3$, $T = 4$, $\zeta = 0$, $\psi = 1$, SNR= $20dB$. The results are averaged over $R = 100$ runs.

4.5 Multiple Measurement Vector Problems with Dynamic Support

The aim of this section is to investigate the performance of the EM-AMP-DCS method. Justin ziniel et al. [ZS13a] have implemented the EM-AMP-DCS model in a Matlab toolbox[9], which we will utilize for all experiments involving EM-AMP-DCS model.

We will again utilize TNMSE and F-measure to quantify the performance. For the EM-AMP-DCS model, we will estimate the support as follows:

$$\forall i \in [n], \forall t \in [T]: \; \hat{s}_i^t = \arg\max_{s_i^t} p(s_i^t \big| \boldsymbol{Y}) \tag{4.20}$$

Performance of EM-AMP-DCS vs. EM-BG-AMP

In this experiment, the EM-AMP-DCS method is compared to the EM-BG-AMP method on problem with slowly changing support. Ideally, the performance of the EM-AMP-DCS model should be compared to other models, which also mode the evolution of the support. But the existence of such models is very limited and therefore we have to settle for the comparison with the EM-BG-AMP model.

We will now describe a series of numerical experiments similar to those, which were carried out for the AMP-MMV model in the last section. The only major difference is that the support of the true solutions $\boldsymbol{X}$ is no longer constant, but now it evolves according to a 2-state discrete Markov chain governed by the sparsity parameter λ and the transitional probability $p_{10} = p(s_i^t = 0|s_i^{t-1} = 1)$.

We will investigate how the TNMSE and F-measure of the EM-AMP-DCS behaves as a function of the undersamplingsratio δ, the degree of sparsity ρ, the signal-to-noise ratio SNR and the transitional probability p_{10}.

In all simulations, the data are generated using an I.I.D. Gaussian forward matrix, where the columns have been scaled to unit ℓ_2-norm with the configuration $n = 1000$, $T = 20$, $\delta = 0.1$, and $\rho = 0.3$. This is the base set up for all the experiments in this subsection. If not specified otherwise, these parameters are used in all the simulations. For each $t = 1, .., T$, the noise variance is scaled to provide a fixed SNR of 20dB. The true solutions $\boldsymbol{X}$ are sampled from the prior

[9] http://www2.ece.ohio-state.edu/~zinielj/dcs/index.html

of the AMP-DCS model with the following hyperparameters:

$$\zeta = 0 \quad, \psi = 1, \quad \alpha = 0.1, \quad \lambda = \rho\delta, \quad \sigma_c^2 = 1, \quad p_{10} = 0.05 \tag{4.21}$$

Suppose the i'th source is active a time t and let L denote the number of consecutive frames, where the i'th source remain active. Since the activity of the sources evolve according to a 2-state Markov chain, it implies that L follows a geometrical distribution, where the corresponding mean and variance are given by $\mathbb{E}[L] = \frac{1}{p_{10}}$ $\mathbb{V}[L] = \frac{1-p_{10}}{(p_{10})^2}$, respectively [PK11]. For the current choice of $p_{10} = 0.05$, we have

$$\mathbb{E}[L] = 20 = T$$
$$\mathbb{V}[L] = 380$$

Using the geometric distribution, we easily obtain the probabilities $p(L < 5) = 0.186$, $p(L < 10) = 0.370$, $p(L < 15) = 0.512$. Therefore, given the high number of sources, i.e. $n = 1000$, the common sparsity assumption is likely not to be fulfilled even when the expected length is equal to the number of measurement vectors.

For a proper comparison, the hyperparameters shared by the two models are initialized to the same values:

$$\zeta = 0 \quad, \psi = 1.5, \quad \lambda = \rho_{se}(\delta) \cdot \delta, \quad \sigma^2 = 0.5 \tag{4.22}$$

The hyperparameters specific to the EM-AMP-DCS models are initialized as follows:

$$\alpha = 0.5, \quad p_{10} = 0.1, \quad \sigma_c^2 = 1.5 \tag{4.23}$$

These values are used in all simulations in the rest of this subsection. The EM-AMP-DCS was configured to perform at least 25 EM iterations, since it is hypothesized that it needs more iterations than the EM-AMP-MMV model due to the increased flexibility. The maximum number of EM iterations for both EM-AMP-DCS and EM-BG-AMP is fixed to 100. The number of AMP iterations for both methods is fixed to 25.

In the first experiment, the undersamplingsratio δ is sampled in the interval from 0.02 to 0.3 and the results are shown in figure 4.27. The advantage of using the EM-AMP-DCS compared to EM-BG-AMP is clearly seen from these figures. For all values of δ, the EM-AMP-DCS algorithm outperforms the SMV approach both in terms of TNMSE and F-measure. It is seen that, the EM-AMP-DCS method needs the undersamplingsratio δ to be larger than 0.1 to recover the true support, while the EM-BG-AMP peaks at $F \approx 0.8$ for even

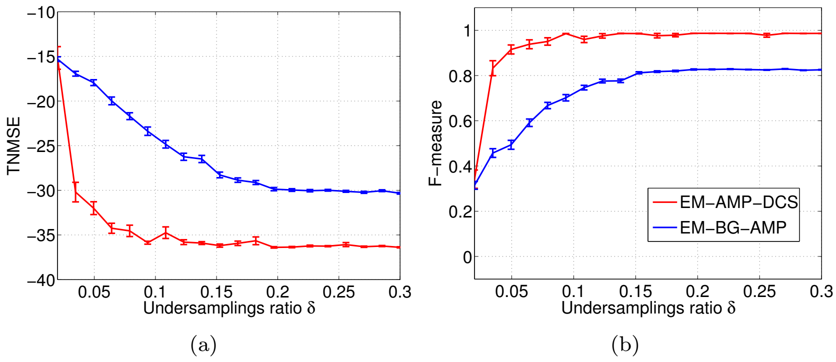


Figure 4.27: Performance vs. undersamplingsratio. The data are generated using an I.I.D. Gaussian forward matrix and using the parameters $n = 1000, \rho = 0.3$, SNR$= 20dB$, $T = 20$. Underlying true signal: $\zeta = 0, \psi = 1, \alpha = 0.9, p_{10} = 0.05$. The results are averaged over $R = 100$ runs.

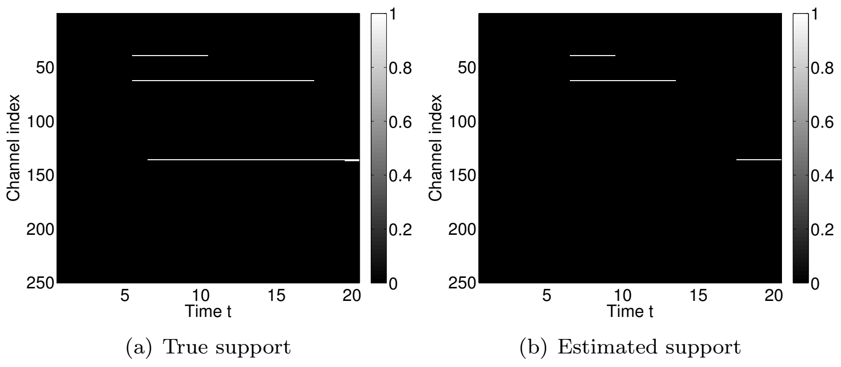


Figure 4.28: The first 250 rows of the true support and the estimated support for EM-AMP-DCS for $\delta = 0.025$, $n = 1000, \rho = 0.3$, SNR$= 20dB$, $T = 20$. Underlying true signal: $\zeta = 0, \psi = 1, \alpha = 0.9, p_{10} = 0.05$.

higher values of δ. The errors bars in the figure indicate a bit more variation in these results compared to the corresponding plots for the AMP-MMV model. This is probably due the high variance of L.

In order to understand the type of errors made by the EM-AMP-DCS algorithm, we will investigate one of the problem instances generated in the above experiment. Figures 4.28(a)-(b) show the first 250 rows of true support and the estimated support, respectively, for one problem instance, where $\delta = 0.025$. In this region, the average F-measure for EM-AMP-DCS is approximately 0.5. Based on these plots, it is seen that the algorithm tends to produce false negatives, when it is not able to recover the support perfectly.

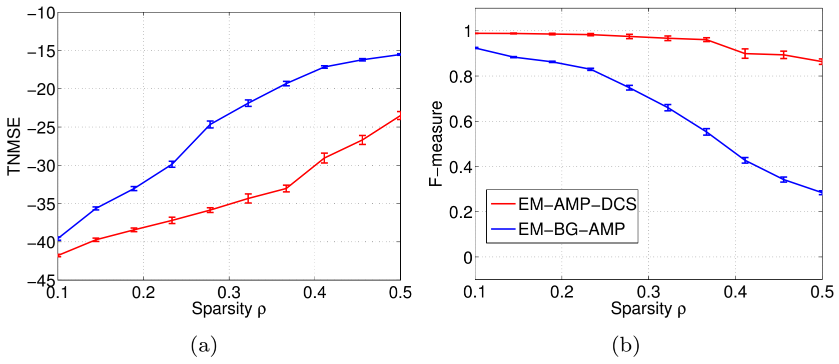
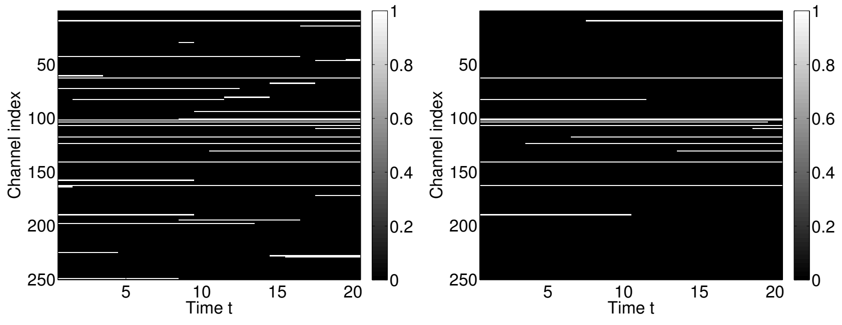


Figure 4.29: Performance vs. sparsity ρ: The data are generated using an I.I.D. Gaussian forward matrix and using the parameters $n = 1000, \delta = 0.1, SNR = 20dB$, $T = 20$. Underlying true signal: $\zeta = 0, \psi = 1, \alpha = 0.9, p_{10} = 0.05$. The results are averaged over $R = 100$ runs.

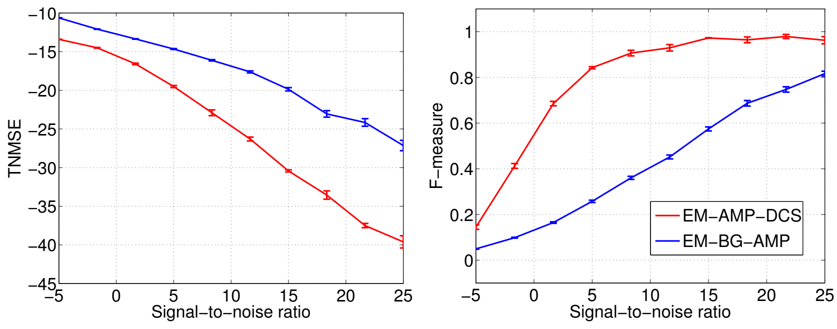
In the next experiment, the sparsity ρ is sampled in the interval from 0.1 to 0.5 and the results are shown in figures 4.29(a)-(b). It is seen that for all values of ρ, the TNMSE error is lower for EM-AMP-DCS than for EM-BG-AMP. Moreover, for small values of ρ, i.e. a small number of non-zero elements in $\boldsymbol{X}$, both algorithms are able to achieve an F-measure of 0.9 or higher. But as the ρ increases, the F-measure for EM-BG-AMP drops, while the F-measure for EM-AMP-DCS is much more stable.

Figure 4.30 shows the first 250 rows of the true support and the estimated support for EM-AMP-DCS for one problem instance, where $\rho = 0.5$. Again, it is seen that the EM-AMP-DCS algorithm produces false negatives, when it is not able to recover the support perfectly.



(a) True support (b) Estimated support

Figure 4.30: The first 250 rows of the true support and the estimated support for EM-AMP-DCS for $\rho = 0.5$. $n = 1000, \delta = 0.1$, SNR$= 20dB$, $T = 20$. Underlying true signal: $\zeta = 0, \psi = 1, \alpha = 0.9, p_{10} = 0.05$.



(a) (b)

Figure 4.31: Performance vs. SNR: The problem data generated using an I.I.D. Gaussian forward matrix and using the parameters $n = 1000, \delta = 0.1, \rho = 0.3$, $T = 20$. Underlying true signal: $\zeta = 0, \psi = 1, \alpha = 0.9, p_{10} = 0.05$. The results are averaged over $R = 100$ runs.

Figures 4.31(a)-(b) show TNMSE and the F-measure as a function of the signal-to-noise ratio is sampled in the interval $-5dB$ to $25dB$. Again, it is seen that the EM-AMP-DCS outperforms the EM-BG-AMP method on both TNMSE and F-measure for all values of the signal-to-noise ratio. But figure (b) also shows that even with $T = 20$, we need at least a signal-to-noise ratio of 10-15 dB to produce decent results.

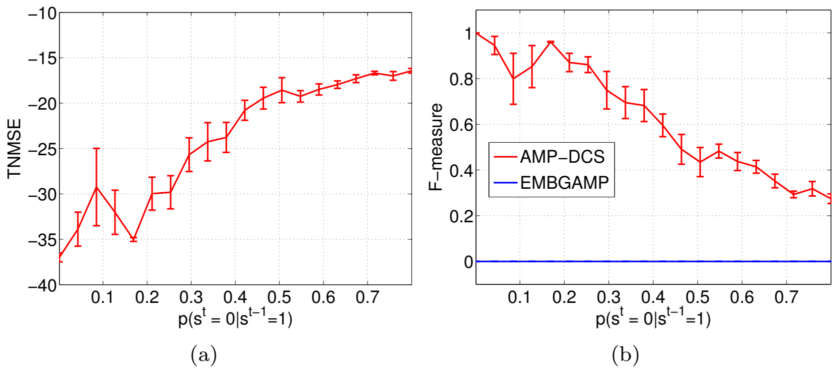


Figure 4.32: Performance vs. p_{10}. The data are generated using an I.I.D. Gaussian forward matrix and using the parameters $n = 1000, \delta = 0.1, \rho = 0.3$, SNR$= 20dB$, $T = 20$. Underlying true signal: $\zeta = 0, \psi = 1, \alpha = 0.9$. The results are averaged over $R = 100$ runs.

In the last experiment in this subsection, the transition probability p_{10} is sampled in the interval from 10^{-3} to 0.8 and the results are shown in figures 4.32(a)-(b). As expected for small values of p_{10}, the EM-AMP-DCS model clearly outperforms the EM-BG-AMP model, but interestingly, for approximate $p_{10} > 0.35$ the EM-BG-AMP method actually outperforms the EM-AMP-DCS method. The performance of the EM-AMP-DCS method was expected to be worst at approximate $p_{10} = 0.5$, where the uncertainty of the Markov model is highest. Furthermore, the performance curves were expected to be approximately symmetric around the point $p_{10} = 0.5$. But this is clearly not the case. However, the observed curves might be caused by the EM algorithm getting stuck in a poor local maxima due to the fact that the hyperparameter $\hat{p}_{10}$ is initialized as $\hat{p}_{10}^0 = 0.1$ independent of the true value of p_{10}.

Thus, we conclude that when the support is changing sufficiently slow, the EM-AMP-DCS model outperforms the EM-BG-AMP in all aspects. But on the other hand, if the support is changing too fast, the EM-BG-AMP model and the SMV approach should be preferred.

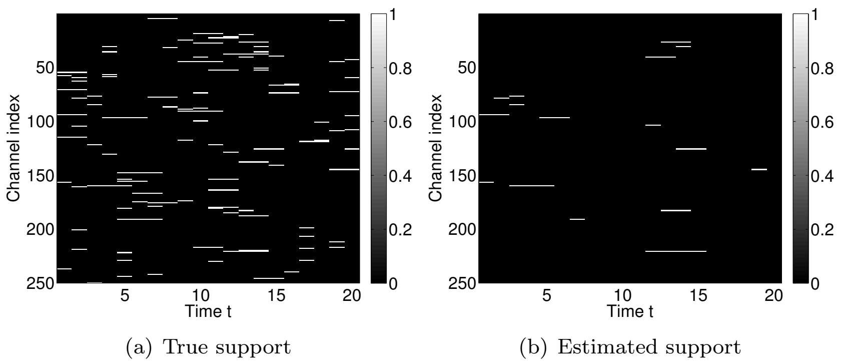


Figure 4.33: The first 250 rows of the true support and the estimated support for EM-AMP-DCS for $\rho = 0.3$. $n = 1000, \delta = 0.1$, SNR$= 20dB$, $T = 20$. Underlying true signal: $\zeta = 0, \psi = 1, \alpha = 0.9, p_{10} = 0.7$.

4.6 EEG Source Localization

The purpose of this very last experiment is to simulate an EEG source localization problem (see appendix D for more details). In particular, an experiment is set up to mimic the configuration of a real EEG inverse problem with $n = 8196$ sources, $m = 128$ measurements, and $T = 100$ measurement vectors. We will simulate a source signals $\boldsymbol{X}$ with an average of 20 active sources, generate the measurements $\boldsymbol{Y}$, and then try to recover the true sources using EM-AMP-DCS and EM-BG-AMP.

The forward matrices in all simulations so far in this thesis have been Gaussian I.I.D. matrices. Gaussian matrices with independent elements are almost surely orthogonal and well-behaved, but this is not the case with real forward matrices. In fact, EEG forward matrices derived from head models are often heavily ill-conditioned. We will therefore perform the entire experiment twice. First, the experiment is performed with an I.I.D. Gaussian forward matrix and then the entire experiment is repeated using a real EEG forward matrix obtained from a head model. This comparison is intended to illustrate and quantify the significance of the forward matrix. The specific real EEG matrix is obtained from OpenMEEG [GPOC10]. The exact same source signals $\boldsymbol{X}$ and noise $\boldsymbol{E}$ will be used for both experiments.

The true source signals $\boldsymbol{X}$ are generated as follows. First the support of the solution $\boldsymbol{S}$ is generated using the 2-state Markov Chain from the prior distribution of AMP-DCS with the following parameters:

$$\lambda = \frac{20}{8196}, \qquad p_{10} = \frac{1}{40}$$

Thus, this particular value of λ correspond to an average of 20 active sources at any given time t. The coefficients of these sources are then generated using sine waves with frequencies sampled uniformly in the interval 5Hz to 30Hz using a sampling frequency of 250Hz. The amplitude of the sine waves are fixed to 1. The measurements are then generated as:

$$\boldsymbol{y}^t = \boldsymbol{A}\boldsymbol{x}_0^t + \boldsymbol{e}_t \qquad \forall t \in [T]$$

The noise is zero-mean, isotropic Gaussian noise, where the variance has been scaled to provide an fixed signal-to-noise ratio of SNR$= 12dB$.

The undersamplingratio δ and sparsity level of this particular configuration corresponds to:

$$\delta = \frac{m}{n} = 0.0156, \qquad\qquad \rho = \delta^{-1}\lambda = 0.1562 \tag{4.24}$$

By comparing these values to the phase transition curve in figure 4.9, it is seen that this particular problem is difficult to solve, especially using the SMV approach.

Due to the dimension of the source matrix ($\boldsymbol{X} \in \mathbb{R}^{8196\times 100}$), it is practically impossible to visualize the entire matrix $\boldsymbol{X}$. Therefore, we only show rows of X, which have active coefficients for at least one value of t. Moreover, for visualization purposes the rows of $\boldsymbol{X}$ are sorted according to the degree of source activity such that the first row corresponds to the source with most active time frames. That is, the rows are sorted according to $\sum_{t=1}^{T} s_i^t$ in descending order. Note, that this does not affect the performance of the algorithms. Figure 4.34(a) shows the true active sources $\boldsymbol{X}$.

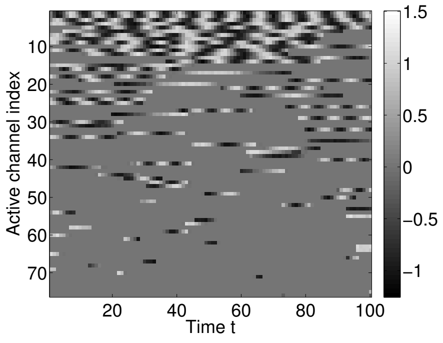
For a proper comparison, the hyperparameters shared by the two models are initialized to the same values:

$$\zeta = 0 \quad, \psi = 1, \quad \lambda = \rho_{se}(\delta)\cdot\delta, \quad \sigma^2 = 0.1 \tag{4.25}$$

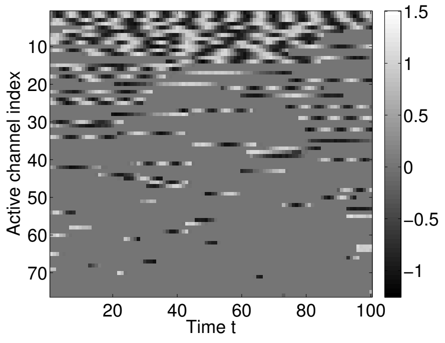
The hyperparameters specific to the EM-AMP-DCS models are initialized as follows:

$$\alpha = 0.1, \quad p_{10} = 0.1, \sigma_c^2 = 1 \tag{4.26}$$

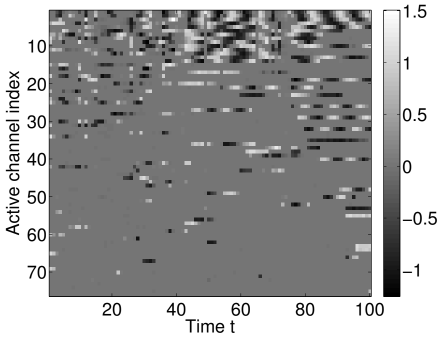
These values are used in all simulations in the rest of this subsection. The EM-AMP-DCS was configured to perform at least 25 EM iterations, since it is hypothesis that it needs more iterations than the EM-AMP-MMV model due to the increased flexibility. The maximum number of EM iterations for both EM-AMP-DCS and EM-BG-AMP is fixed to 100. The number of AMP iterations is fixed to 25 for both methods.



(a) True sources



(b) EM-AMP-DCS



(c) EM-BG-AMP

Figure 4.34: Visualization of the active sources, i.e. sources which are active for at least one value of t. $F_{DCS} = 0.996, F_{SMV} = 0.284$, $TNMSE_{DCS} = -44.954dB, TNMSE_{SMV} = -20.941dB$

We now consider the task of reconstructing the sources based on the measurements generated using the I.I.D. Gaussian forward matrix $\boldsymbol{A}$, where the columns have been scaled to unit ℓ_2-norm. Figures 4.34(b) and (c) show the signals recovered using EM-AMP-DCS and EM-BG-AMP, respectively. By comparing figure (a), (b), and (c), it is not difficult to see that the reconstruction using EM-AMP-DCS is more accurate than the reconstruction using EM-BG-AMP.

We now take a closer look on three of the reconstructed active sources, see figure 4.35. The figures show three randomly selected active sources and their

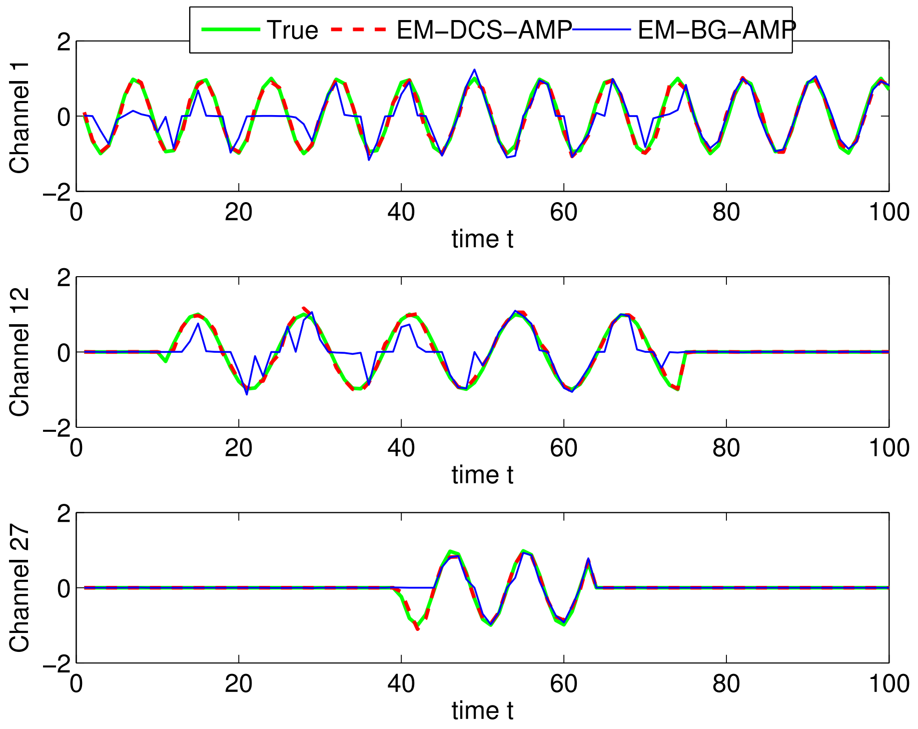


Figure 4.35: Example of the reconstruction of three active sources

corresponding reconstructions. Here it is seen how EM-AMP-DCS is capable of turning sources on and off as needed, while EM-BG-AMP only succeed to recover a small portion of the true signal.

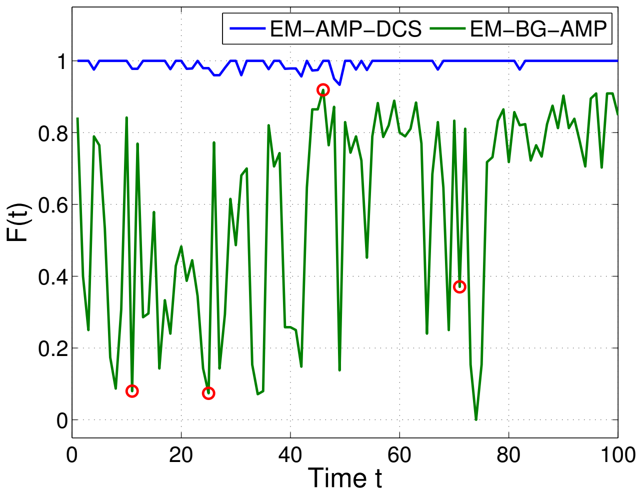
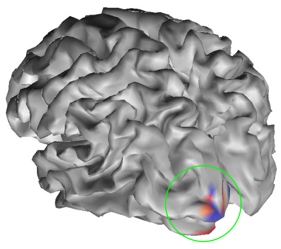


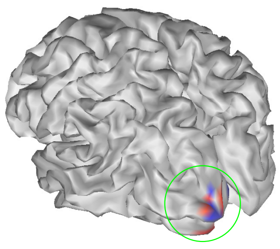
Figure 4.36: F-measure as a function of time for the EEG example

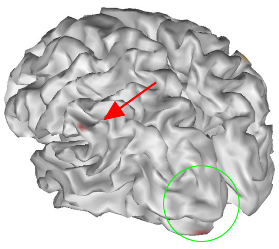
Figure 4.36 shows the F-measure for both methods as a function of time t. Consistent with figure 4.35, it is seen that EM-AMP-DCS provides a decent reconstruction for most values of t, while the F-measure for EM-BG-AMP fluctuates heavily around $F \approx 0.5$.

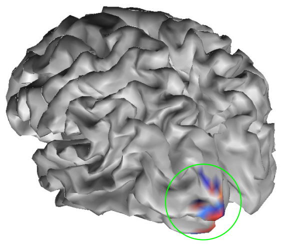
Using a 3D model of the brain, we will now visualize how the reconstructed sources look at the time steps indicated by the red circles on figure 4.36. These visualizations are shown in figure 4.37, which contain four rows and three columns. The first row corresponds to the time $t_1 = 11$, the second row correspond to the time $t_2 = 25$, the third row correspond to the time $t_3 = 46$ and the fourth row correspond to the time $t_4 = 71$. The left-most column shows the true source for the given value of t, while the center column shows the sources estimated by EM-AMP-DCS and the right-most column shows the sources estimated by EM-BG-AMP. The green circles indicate the areas, which should contain the true active sources and the red arrows indicate false positives. It is seen that the EM-AMP-DCS algorithm nicely reconstructs the location of the true sources, while the EM-BG-AMP algorithm only succeed to reconstruct the true sources in one of the four time steps.

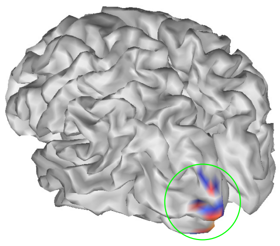
The entire experiment is now repeated with the forward matrix obtained from OpenMEEG. The forward matrix has been scaled such that the columns have

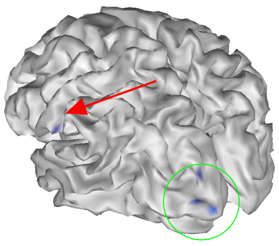

(a) True sources

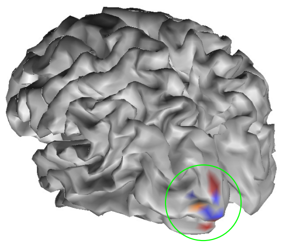

(b) EM-DCS-AMP

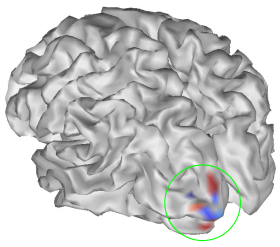

(c) EM-BG-AMP

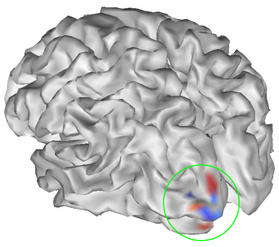

(d) True sources

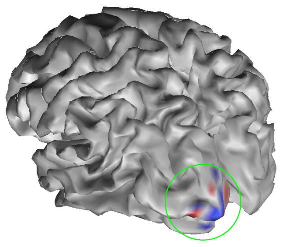

(e) EM-DCS-AMP

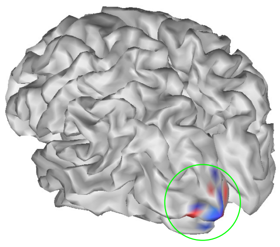

(f) EM-BG-AMP

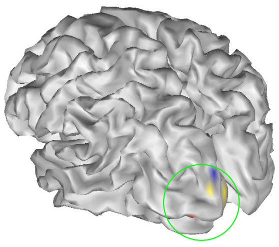

(g) True sources

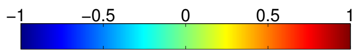

(h) EM-DCS-AMP


(i) EM-BG-AMP


(j) True sources


(k) EM-DCS-AMP


(l) EM-BG-AMP



(m) Colorbar

Figure 4.37: Visualization of the sources obtained using the Gaussian matrix. The four rows correspond to different values of $t = 11, 25, 46, 71$. The left column correspond to the true sources, the mid column correspond to the sources obtained using EM-AMP-DCS and the right column correspond to the sources obtained using EM-BG-AMP.

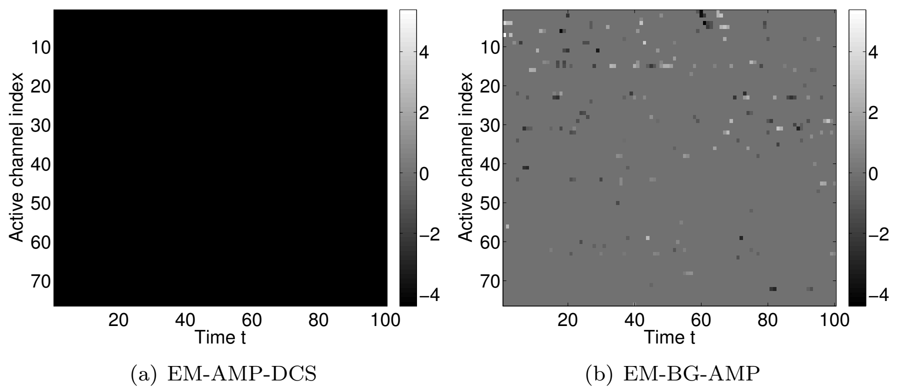


Figure 4.38: Visualization of the active sources, i.e. sources which are active for at least one value of t. $F_{SMV} = 0.091, TNMSE_{SMV} = 0.0149dB$

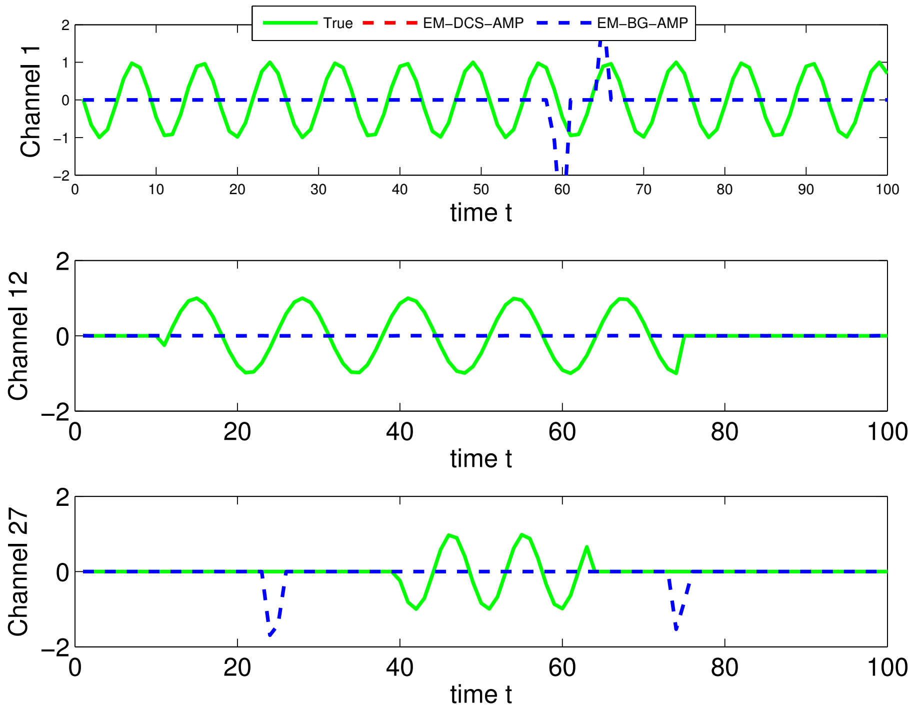


Figure 4.39: Real EEG forward matrix: Example of the reconstruction of three active sources

unit ℓ_2-norm, before the measurements were generated. Figures 4.38 and 4.39 show the reconstructed sources using both EM-AMP-DCS and EM-BG-AMP. It is seen that the two estimated solutions are very different from the solutions obtained from the Gaussian forward matrix. Here, the EM-AMP-DCS simply fails to due numerical overflow. After the first EM iteration, most of the estimated values in $\hat{\boldsymbol{X}}$ are of order 10^{43}. Thus, despite both algorithms have shown excellent recovery capabilities, they are both virtually useless when applied to a real forward matrix. Table 4.1 compared the results for the two experiments.

The coherence and the condition number for the two forward matrices are:

$$\begin{aligned} \text{Cond}(\boldsymbol{A}_{\text{real}}) &= 3.8060 \cdot 10^{15} \qquad & \text{Coherence}(\boldsymbol{A}_{\text{real}}) &= 0.9998 \\ \text{Cond}(\boldsymbol{A}_{\text{gauss}}) &= 1.2727 \qquad & \text{Coherence}(\boldsymbol{A}_{\text{gauss}}) &= 0.4779 \end{aligned}$$

The coherence is computed as: $\max_{i,j} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle|$, where the $\langle \cdot, \cdot \rangle$ is the Euclidean inner product and $\boldsymbol{a}_i$ are the i'th column of $\boldsymbol{A}$. The condition number shows that the real EEG forward matrix is indeed severely ill-conditioned and the coherence measure imply that there are at least two columns in the real EEG forward matrix, that are almost parallel.

Method	**TNMSE** [dB]	**F**	**TNMSE** [dB]	**F**
	Gaussian $\boldsymbol{A}$		Real EEG $\boldsymbol{A}$	
AMP-DCS	-44.954	0.996	-	-
EM-BG-AMP	-20.941	0.284	0.0149	0.091

Table 4.1: Comparison of the results from the experiment with a real EEG forward matrix vs. the experiment with a I.I.D Gaussian forward matrix.

This experiment emphasizes the importance of the forward matrix. It is clearly seen that the properties of the forward matrix can have a crucial effect of the recovery performance. Moreover, this experiment suggests that we should not have too high hopes for solving the EEG source localization problem using this particular configuration. However, there are several ways to reduce the difficulty of the problem. The signal to noise ratio of raw EEG signals are often very low, i.e. 0-12dB. But the effective SNR can be significantly increased by a series of preprocessing steps including simple band-pass filtering [NS06], artefacts removal [ZRZ+08] and independent component analysis [ZWF01]. Furthermore, using a basis function approach can effectively reduce the dimension of the problem, since the number of basis functions is usually smaller than the number of sources. In the context of source localization, both spatial and temporal basis functions are used [BG97, SAS+13].

This ends the chapter on numerical experiments conducted in this thesis.

Chapter 5

Conclusion

This thesis has examined message passing-based methods for the purpose of solving underdetermined linear inverse problems. More specifically, problems where the number of measurements m is much smaller than the number of equations n. The approximate message passing algorithm (AMP) has been derived by applying a series of approximations to the sum-product algorithm. Central to these approximations is the assumption of the large system limit, i.e $m, n \to \infty$ for $m/n \to \delta$. Despite this assumption, it has been shown empirically that the AMP algorithm is capable of solving linear systems as small as $n = 500$ for an undersamplingsratio of $m/n = 0.5$.

The generalized approximate message passing algorithm (GAMP), which is a Bayesian generalization of the AMP algorithm, has also been derived. The GAMP algorithm allows the use of a broad class of prior and noise distributions for both MMSE and MAP estimation. It has been shown analytically and verified experimentally that the AMP algorithm indeed is as a special case of GAMP algorithm. In fact, the GAMP algorithm correspond to the AMP algorithm, if the prior distribution is chosen to be a Laplace distribution and the noise distribution is chosen to be Gaussian.

The EM-BG-AMP algorithm is derived for solving the linear inverse problem in the SMV formulation by combining the GAMP algorithm with a Gaussian noise distribution and a Bernoulli-Gaussian prior distribution. This algorithm is complemented by an Expectation-Maximization (EM) scheme for automatic tuning of the hyperparameters. Moreover, the EM-BG-AMP algorithm provides both MMSE estimates of the desired solution as well as MAP estimates of the underlying support.

The performance of AMP and EM-BG-AMP have been quantified in terms of the sparsity undersampling trade-off using empirical phase transition curves for noiseless problems with Gaussian forward matrices. These curves show that the EM-BG-AMP method outperforms both AMP, Bayesian VG and FOCUSS in terms of the sparsity undersampling trade-off. The estimated curve for AMP largely agrees with the theoretical asymptotic curve for ℓ_1 minimization as predicted by the state evolution formalism, except for undersamplingratios smaller than approximately 0.1.

The CPU run times for these four methods were also measured and the following order was observed: AMP $<$ EM-BG-AMP $<$ FOCUSS $<$ Bayesian VG, where AMP is the fastest method. The analytical computational complexities of these methods agree with the observed order.

Additionally, using normalized mean square error and F-measure as performance measures, it is demonstrated that the EM-BG-AMP algorithm outperforms the Variational Garrote and LASSO methods in terms of undersamplingsratio, sparsity level and signal-to-noise ratio. Therefore, it can be concluded that the EM-BG-AMP algorithm provides superior recovery performance in both the noiseless and noisy setting.

The EM-AMP-MMV algorithm is derived by extending the EM-BG-AMP algorithm and the accompanying EM algorithm to the multiple measurement vector (MMV) formulation using the common sparsity assumption. This algorithm also models the temporal correlation of the coefficients of the solution using a first order stationary Gauss-Markov process. The benefits of the MMV formulation compared to the SMV formulation is demonstrated through a numerical experiment using synthetic data. Problems, which were unsolvable in the SMV formulation, became solvable in the MMV formulation, when the number of measurement vectors was increased.

The EM-AMP-MMV method is compared to M-FOCUSS and the state-of-the-art Bayesian methods, M-SBL and TM-SBL, in a systematic manner. The four methods are compared in terms of undersamplingsratio, sparsity, signal-to-noise-ratio, number of measurement vectors and run time. The main conclusion from these experiments is that the recovery performance of EM-AMP-MMV is comparable to the recovery performance of the state-of-the-art methods, but the computational complexity of EM-AMP-MMV is much lower than the other methods. In fact, the EM-AMP-MMV algorithm scales linearly in all problem dimensions. The linear complexity allows the EM-AMP-MMV algorithm to be applied to large-scale problems, where methods like TM-SBL and M-SBL are simply infeasible. However, for problems with small sample sizes, the recovery performance of TM-SBL and M-SBL were slightly better than EM-AMP-MMV, but at the cost of a considerably increase in run time. Therefore, we conclude that for most MMV problems the EM-AMP-MMV algorithm is preferable. Especially, for problems where recovery performance and speed are equally important.

The last model considered in this thesis is the EM-AMP-DCS model. This model extends the EM-AMP-MMV model by relaxing the assumption of a constant sparsity pattern to a slowly changing sparsity pattern. The EM-AMP-DCS algorithm models the slowly changing support of the underlying solution using binary Markov chains. The hyperparameters of this model, including the transitional probabilities for the Markov chain, are estimated from the data using an EM update scheme. Using numerical simulations, it was shown that the EM-AMP-DCS model is superior to the EM-BG-AMP model in terms of undersamplingsratio, sparsity and signal to noise ratio given the support is changing sufficient slow. However, the SMV approach, i.e. EM-BG-AMP, was found to be superior, if the support of the underlying solution is changing too fast.

The EM-BG-AMP and EM-AMP-DCS algorithms were applied to a severly underdetermined EEG source localization problem with synthetic sources. Using a configuration with 8196 sources, 128 measurements, 100 measurement vectors, 20 active harmonic sources and a Gaussian forward matrix, it was shown that the EM-AMP-DCS algorithm was able to reconstruct the underlying sources almost perfectly, while the EM-BG-AMP method was only able to recover a small portion of the true sources. The entire experiment was repeated using severely ill-conditioned EEG forward matrix extracted from a head model, and it was shown that neither of the algorithms were able to recover the true sources.

Overall, approximate message passing-based methods for both the SMV and MMV formulation have been shown to yield high performance in terms of the undersampling sparsity trade-off, while maintaining linear computational complexity.

5.1 Future work

This thesis is concluded with a list of interesting topics to examine in the future:

- The EM-AMP-DCS algorithm was not able to solve the simulated EEG source localization problem, probably due to a combination of a strongly illconditioned forward matrix and a highly underdetermined system. It would therefore be interesting to combine the AMP algorithms with a basis function approach to reduce the effective dimension of the problem.

- The phase transition curves for single measurement vector problems constitute a systematic way of comparing the performance of different reconstruction algorithms. To the author's best knowledge, there exist no systematic way of comparing recovery performance for MMV problems. But it would indeed be fruitful to have a principled way of comparing performance for MMV algorithm, like "phase diagrams for MMV problems".
- The AMP-MMV and AMP-DCS models offer the possibility of causal filtering of MMV signals. The recovery performance for such an approach is expected to be suboptimal compared to the smoothing approach, but the reduced computational complexity has large potential in real-time and online applications.

Appendix A

AMP

A.1 Gaussian Multiplication Rule

This appendix proofs the Gaussian multiplication rule, i.e. the multiplication of the probability density function of two independent univariate Gaussian distributions over the same variable x:

$$\mathcal{N}\left(x\,|a,A\right)\mathcal{N}\left(x\,|b,B\right)=\mathcal{N}\left(x\,|\,\frac{\frac{a}{A}+\frac{b}{B}}{\frac{1}{A}+\frac{1}{B}},\frac{1}{\frac{1}{A}+\frac{1}{B}}\right)\mathcal{N}\left(0\,|a-b,A+B\right) \tag{A.1}$$

We start by inserting the definition of the densities:

$$\begin{aligned}\mathcal{N}\left(x\,|a,A\right)\mathcal{N}\left(x\,|b,B\right)&=\frac{1}{\sqrt{2\pi A}}\exp\left(-\frac{(x-a)^2}{2A}\right)\frac{1}{\sqrt{2\pi B}}\exp\left(-\frac{(x-b)^2}{2B}\right)\\&=\frac{1}{\sqrt{2\pi A}}\frac{1}{\sqrt{2\pi B}}\exp\left(-\frac{(x-a)^2}{2A}-\frac{(x-b)^2}{2B}\right)\end{aligned}$$

By expanding the argument to the exponential function and reducing, we get:

$$\begin{aligned}-\frac{(x-a)^2}{2A}-\frac{(x-b)^2}{2B}&=-\frac{1}{2A}\left(x^2+a^2-2xa^2\right)-\frac{1}{2B}\left(x^2+b^2-2xb^2\right)\\&=-\frac{x^2}{2}\left(\frac{1}{A}+\frac{1}{B}\right)+x\left(\frac{a}{A}+\frac{b}{B}\right)-\frac{a^2}{2A}-\frac{b^2}{2B}\end{aligned} \tag{A.2}$$

Now tedious, but straightforward manipulations of the two last terms lead to:

$$\begin{aligned}-\frac{a^2}{2A}-\frac{b^2}{2B}&=-\frac{1}{2}\frac{a^2B+b^2A}{AB}\\&=-\frac{1}{2}\frac{a^2B+b^2A}{AB}\frac{A+B}{A+B}\\&=-\frac{1}{2}\frac{a^2B+b^2A}{A+B}\frac{A+B}{AB}\\&=-\frac{1}{2}\frac{a^2AB+b^2A^2+a^2B^2+b^2AB}{A+B}\frac{1}{AB}\\&=-\frac{1}{2}\frac{a^2+b^2+\frac{B}{A}a^2+\frac{A}{B}b^2}{A+B}\\&=-\frac{1}{2}\frac{a^2+b^2+\frac{B}{A}a^2+\frac{A}{B}b^2+2ab-2ab}{A+B}\\&=-\frac{1}{2}\frac{(b-a)^2+\frac{B}{A}a^2+\frac{A}{B}b^2+2ab}{A+B}\\&=-\frac{1}{2}\frac{\frac{B}{A}a^2+\frac{A}{B}b^2+2ab}{A+B}-\frac{1}{2}\frac{(b-a)^2}{A+B}\\&=-\frac{1}{2}\frac{\frac{a^2}{A^2}+\frac{b^2}{B^2}+\frac{2ab}{AB}}{\frac{1}{A}+\frac{1}{B}}-\frac{1}{2}\frac{(b-a)^2}{A+B}\\&=-\frac{1}{2}\frac{\left(\frac{a}{A}+\frac{b}{B}\right)^2}{\frac{1}{A}+\frac{1}{B}}-\frac{1}{2}\frac{(0-(a-b))^2}{A+B}\end{aligned} \tag{A.3}$$

We can now plug the above result in eq. (A.3) back into eq. (A.2):

$$\begin{aligned}-\frac{(x-a)^2}{2A}-\frac{(x-b)^2}{2B}=&-\frac{x^2}{2}\left(\frac{1}{A}+\frac{1}{B}\right)+x\left(\frac{a}{A}+\frac{b}{B}\right)\\&-\frac{1}{2}\frac{\left(\frac{a}{A}+\frac{b}{B}\right)^2}{\frac{1}{A}+\frac{1}{B}}-\frac{1}{2}\frac{(0-(a-b))^2}{A+B}\end{aligned} \tag{A.4}$$

Now consider the argument of the exponent of a third Gaussian distribution over x with mean μ and variance σ

$$-\frac{(x-\mu)^2}{2\sigma} = -\frac{x^2}{2}\frac{1}{\sigma^2} + x\frac{\mu}{\sigma} - \mu^2\frac{1}{2\sigma^2}$$

Then by comparing coefficients for x^2, we get

$$\frac{1}{\sigma^2} = \frac{1}{A} + \frac{1}{B} \qquad \Longleftrightarrow \qquad \sigma^2 = \frac{1}{\frac{1}{A}+\frac{1}{B}}$$

and the same for the coefficients for x and using the result for σ^2 yields:

$$\frac{\mu}{\sigma^2} = \frac{a}{A} + \frac{b}{B} \qquad \Longleftrightarrow \qquad \mu = \left(\frac{a}{A}+\frac{b}{B}\right)\cdot\sigma^2 = \frac{\frac{a}{A}+\frac{b}{B}}{\frac{1}{A}+\frac{1}{B}}$$

Therefore, we get

$$\mathcal{N}(x\,|a,A)\,\mathcal{N}(x\,|b,B) = \frac{1}{\sqrt{2\pi A}}\frac{1}{\sqrt{2\pi B}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\exp\left(-\frac{1}{2}\frac{(0-(a-b))^2}{A+B}\right)$$

Then the normalization terms are rewritten:

$$\begin{aligned}
\frac{1}{\sqrt{2\pi A}}\frac{1}{\sqrt{2\pi B}} &= \frac{1}{\sqrt{(2\pi)^2 AB}} \\
&= \frac{1}{\sqrt{(2\pi)^2 AB\frac{A+B}{A+B}}} \\
&= \frac{1}{\sqrt{(2\pi)^2 (A+B)\frac{AB}{A+B}}} \\
&= \frac{1}{\sqrt{(2\pi)^2 (A+B)\frac{1}{\frac{1}{A}+\frac{1}{B}}}} \\
&= \frac{1}{\sqrt{2\pi(A+B)}}\frac{1}{\sqrt{2\pi\sigma^2}} \qquad \text{(A.5)}
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\mathcal{N}(x\,|a,A)\,\mathcal{N}(x\,|b,B) = \\
&\quad\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\frac{1}{\sqrt{2\pi(A+B)}}\exp\left(-\frac{1}{2}\frac{(0-(a-b))^2}{A+B}\right) \\
&\quad= \mathcal{N}\left(x\,|\frac{\frac{a}{A}+\frac{b}{B}}{\frac{1}{A}+\frac{1}{B}}, A\right)\mathcal{N}(0\,|a-b, A+B)\,, \qquad \text{(A.6)}
\end{aligned}$$

which is the desired result.

A.2 Application of the Berry-Esseen Theorem

The Berry-Esseen theorem is variant of the central limit therem and it is stated and proved in appendix A in [DMM10]. It implies the following for Z and $h_{x_i}(z)$ as defined in eq. (2.45):

$$|\mathbb{E}\left[h_{x_i}(Z)\right] - \mathbb{E}\left[h_{x_i}(W)\right]| \le ||h'||_\infty \frac{C_t}{n^{\frac{1}{2}}\left(\hat{\tau}^t_{a\to i}\right)^{\frac{3}{2}}} = \frac{C'_t}{n^{\frac{1}{2}}\left(\hat{\tau}^t_{a\to i}\right)^{\frac{3}{2}}} \tag{A.7}$$

To use this result, consider:

$$\left|\mu^t_{a\to i}(x_i) - \mathbb{E}\left[h_{x_i}(W)\right]\right| = \left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right]}{\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i} - \frac{\mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i}\right|$$

We can now rewrite the right hand side as follows. First the terms are put on a common denominator:

$$\begin{aligned}&\left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right]}{\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i} - \frac{\mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i}\right| \\ &= \left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right]\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i - \mathbb{E}\left[h_{x_i}(W)\right]\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i}{\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i \int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i}\right|\end{aligned}$$

We can now add and subtract the term $\mathbb{E}\left[h_{x_i}(W)\right]\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i$ in the nominator and rearrange to get:

$$\begin{aligned}&\left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i} - \frac{\mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i}\right| \\ &= \left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right] - \mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i} + \frac{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i - \int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i \int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i}\mathbb{E}\left[h_{x_i}(W)\right]\right|\end{aligned}$$

Applying the triangle inequality

$$\begin{aligned}&\left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i} - \frac{\mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i}\right| \\ &\le \left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right] - \mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i}\right| + \left|\frac{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i - \int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i \int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i}\mathbb{E}\left[h_{x_i}(W)\right]\right|\end{aligned}$$

Since the two normalization integrals integrate to the same, the second term is equal to 0 and we get

$$\left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right]}{\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i} - \frac{\mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(W)\right]\,\mathrm{d}x_i}\right| \le \left|\frac{\mathbb{E}\left[h_{x_i}(Z)\right] - \mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(Z)\right]\,\mathrm{d}x_i}\right|$$

Finally, taking the supremum and applying the Berry-Esseen theorem yields

$$\begin{aligned}\sup_{x_i} \left|\mu^t_{a\to i}(x_i) - \mathbb{E}\left[h_{x_i}(W)\right]\right| &\leq \sup_{x_i} \left| \frac{\mathbb{E}\left[h_{x_i}(Z)\right] - \mathbb{E}\left[h_{x_i}(W)\right]}{\int \mathbb{E}\left[h_{x_i}(Z)\right] \,\mathrm{d}x_i} \right| \\ &\leq \frac{C_t}{n^{\frac{1}{2}} \left(\hat{\tau}^t_{a\to i}\right)^{\frac{3}{2}}} \qquad \text{(A.8)}\end{aligned}$$

Appendix B

GAMP

B.1 Solving the Least Squares Problem for GAMP

The purpose of this appendix is to describe the steps in the solution of the optimization problem given by:

$$J = \min_{\boldsymbol{x}} \sum_{j \neq i} \frac{1}{2\tau_j^x} \left(x_j - \hat{x}_{j \to a}^k \right)^2 \quad \text{subject to} \quad z_a = A_{ai} x_i + \sum_{j \neq i} A_{aj} x_j, \tag{B.1}$$

where x_i and z_a are fixed. This is recognized as a least squares problem with an equality constraint and can thus be solved by introducing Lagrange multipliers and forming the Lagrangian function [NW06]:

$$f(\boldsymbol{x}, \lambda) = \sum_{j \neq i} \frac{1}{2\tau_j^x} \left(x_j - \hat{x}_{j \to a}^k \right)^2 + \lambda \left(z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} x_j \right) \tag{B.2}$$

Computing partial derivative w.r.t. x_k equating to zero yields:

$$\begin{aligned} & \frac{\partial}{\partial x_k} f(\boldsymbol{x}, \lambda) = \frac{1}{\tau_k^x} x_k - \frac{1}{\tau_k^x} \hat{x}_{k \to a}^k - \lambda A_{ak} = 0 \\ \iff \quad & x_k = \hat{x}_{k \to a}^k + \lambda \tau_k^x A_{ak} \end{aligned} \tag{B.3}$$

And the same for λ:

$$\frac{\partial}{\partial \lambda} f(\boldsymbol{x}, \lambda) = z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} x_j = 0 \tag{B.4}$$

Substituting eq. (B.3) into the sum in eq. (B.4) and expanding:

$$\begin{aligned} \frac{\partial}{\partial \lambda} f &= z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} \left(\hat{x}_{j \to a}^k + \lambda \tau_j^x A_{aj} \right) = 0 \\ &= z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} \hat{x}_{j \to a}^k - \lambda \sum_{j \neq i} A_{aj}^2 \tau_j^x = 0 \end{aligned} \tag{B.5}$$

Solving for λ yields:

$$\begin{aligned} & z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} \hat{x}_{j \to a}^k - \lambda \sum_{j \neq i} A_{aj}^2 \tau_j^x = 0 \\ \iff \quad & z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} \hat{x}_{j \to a}^k = \lambda \sum_{j \neq i} A_{aj}^2 \tau_j^x \\ \iff \quad & \lambda = \frac{z_a - A_{ai} x_i - \sum_{j \neq i} A_{aj} \hat{x}_{j \to a}^k}{\sum_{j \neq i} A_{aj}^2 \tau_j^x} \end{aligned} \tag{B.6}$$

Now we substitute eq. (B.3) into (B.1):

$$J = \sum_{j \neq i} \frac{1}{2\tau_j^x} \left(\hat{x}_{j\to a}^k + \lambda \tau_j^x A_{aj} - \hat{x}_{j\to a}^k\right)^2$$

and substituting the expression for λ in eq. (B.6) into this expression and rearranging

$$\begin{aligned}
J &= \sum_{j \neq i} \frac{1}{2\tau_j^x} \left(\hat{x}_{j\to a}^k + \tau_j^x A_{aj} \frac{z_a - A_{ai}x_i - \sum_{j\neq i} A_{aj}\hat{x}_{j\to a}^k}{\sum_{j\neq i} A_{aj}^2 \tau_j^x} - \hat{x}_{j\to a}^k\right)^2 \\
&= \sum_{j \neq i} \frac{1}{2\tau_j^x} \left(\tau_j^x A_{aj} \frac{z_a - A_{ai}x_i - \sum_{j\neq i} A_{aj}\hat{x}_{j\to a}^k}{\sum_{j\neq i} A_{aj}^2 \tau_j^x}\right)^2 \\
&= \sum_{j \neq i} \frac{1}{2\tau_j^x} \left(\tau_j^x\right)^2 A_{aj}^2 \left(\frac{z_a - A_{ai}x_i - \sum_{j\neq i} A_{aj}\hat{x}_{j\to a}^k}{\sum_{j\neq i} A_{aj}^2 \tau_j^x}\right)^2 \\
&= \frac{1}{2} \sum_{j \neq i} \tau_j^x A_{aj}^2 \frac{1}{\left(\sum_{j\neq i} A_{aj}^2 \tau_j^x\right)^2} \left(z_a - A_{ai}x_i - \sum_{j\neq i} A_{aj}\hat{x}_{j\to a}^k\right)^2
\end{aligned}$$

Changing summation index for two of the sums on the r.h.s to avoid mixing the indices:

$$J = \frac{1}{2} \sum_{j \neq i} A_{aj}^2 \tau_j^x \frac{1}{\left(\sum_{r\neq i} A_{ar}^2 \tau_r^x\right)^2} \left(z_a - A_{ai}x_i - \sum_{r\neq i} A_{ar}\hat{x}_{r\to a}^k\right)^2$$

Now we recognize that the three terms in the parenthesis does *not* depend on index j and can therefore be moved outside the sum:

$$J = \frac{1}{2} \left(z_a - A_{ai}x_i - \sum_{r\neq i} A_{ar}\hat{x}_{r\to a}^k\right)^2 \frac{\sum_{j\neq i} A_{aj}^2 \tau_j^x}{\left(\sum_{r\neq i} A_{ar}^2 \tau_r^x\right)^2}$$

Cancelling the sums of $A_{aj}^2 \tau_j^x$ and rearranging yields:

$$\begin{aligned}
J &= \frac{1}{2} \left(z_a - A_{ai}x_i - \sum_{r\neq i} A_{ar}\hat{x}_{r\to a}^k\right)^2 \frac{1}{\sum_{r\neq i} A_{ar}^2 \tau_r^x} \\
&= \frac{1}{2} \frac{1}{\sum_{r\neq i} A_{ar}^2 \tau_r^x} \left(z_a - A_{ai}x_i - \sum_{r\neq i} A_{ar}\hat{x}_{r\to a}^k\right)^2,
\end{aligned}$$

which is the solution to the least squares problem in eq. (B.1).

B.2 AMP as a Special Case of GAMP

The purpose of this section is to show the relation between the AMP algorithm and the GAMP2 algorithm with specific choice of distributions. The update equations from AMP0 is:

$$\begin{aligned}
\boldsymbol{x}^{t+1} &= \eta\left(\boldsymbol{A}^T\boldsymbol{z}^t + \boldsymbol{x}^t; \tau^t\right)\\
\boldsymbol{z}^t &= \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^t + \frac{1}{\delta}\boldsymbol{z}^{t-1}\left\langle \eta'\left(\boldsymbol{A}^T\boldsymbol{z}^{t-1} + x^{t-1}; \tau^{t-1}\right)\right\rangle\\
\tau^t &= \frac{\tau}{\delta}\left\langle \eta'\left(\boldsymbol{A}^T\boldsymbol{z}^{t-1} + \boldsymbol{x}^t; \tau^{t-1}\right)\right\rangle
\end{aligned}$$

We will now argue that the GAMP2 algorithm with scalar variances and for specific choices of the input and output distribution is closely related to the AMP0 algorithm.

We assume that the columns of $\boldsymbol{A}$ have unit ℓ_2 norm. Furthermore, we assume the prior distribution is a Laplace distribution with parameter β and the output noise is zero-mean Gaussian noise with variance $\sigma^2 = \frac{1}{\beta}$. Therefore, we have

$$\begin{aligned}
f_{in}\left(x, q\right) &= \ln p\left(x|q\right) = \ln \text{Laplace}(x;\beta) = \ln\frac{\beta}{2} - \beta\left|x\right|\\
f_{out}\left(z, y\right) &= \ln p\left(y|z\right) = \ln \mathcal{N}\left(y; z, \sigma^2\right) = -\frac{1}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\left(y-z\right)^2
\end{aligned}$$

We can now compute the output threshold scalar function g_{out}:

$$\begin{aligned}
g_{out}\left(\hat{p}, y, \tau^p\right) &\equiv \frac{1}{\tau^p}\left(\arg\max_z\left\{f_{out}\left(z,y\right) - \frac{1}{2\tau^p}\left(z-\hat{p}\right)^2\right\} - \hat{p}\right)\\
&= \frac{1}{\tau^p}\left(\arg\max_z\left\{-\frac{1}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\left(y-z\right)^2 - \frac{1}{2\tau^p}\left(z-\hat{p}\right)^2\right\} - \hat{p}\right)\\
&= \frac{1}{\tau^p}\left(\arg\max_z\left\{-\frac{1}{2\sigma^2}\left(y-z\right)^2 - \frac{1}{2\tau^p}\left(z-\hat{p}\right)^2\right\} - \hat{p}\right)
\end{aligned}$$

We now take the derivative of the inner part, setting it equal to 0 and solve for z:

$$\begin{aligned}
& \frac{\partial}{\partial z} = \frac{1}{\sigma^2}\left(y-z\right) - \frac{1}{\tau^p}\left(z-\hat{p}\right) = 0\\
\iff\quad & -z\left(\frac{1}{\sigma^2} + \frac{1}{\tau^p}\right) + \frac{1}{\sigma^2}y + \frac{1}{\tau^p}\hat{p} = 0\\
\iff\quad & z\left(\frac{1}{\sigma^2} + \frac{1}{\tau^p}\right) = \frac{1}{\sigma^2}y + \frac{1}{\tau^p}\hat{p}\\
\iff\quad & z = \frac{\frac{1}{\sigma^2}y + \frac{1}{\tau^p}\hat{p}}{\frac{1}{\sigma^2} + \frac{1}{\tau^p}} = \frac{\tau^p y + \sigma^2\hat{p}}{\tau^p + \sigma^2}
\end{aligned}$$

By substituting back, we get:

$$\begin{aligned} g_{out}\left(\hat{p}, y, \tau^p\right) &= \frac{1}{\tau^p}\left(\frac{\tau^p y + \sigma^2 \hat{p}}{\tau^p + \sigma^2} - \hat{p}\right) \\ &= \frac{y - \hat{p}}{\tau^p + \sigma^2} \end{aligned}$$

and

$$-\frac{\partial}{\partial \hat{p}} g_{out}\left(\hat{p}, y, \tau^p\right) = \frac{1}{\tau^p + \sigma^2}$$

Next, we consider the variance parameters τ^p, τ^r, τ^x and τ^s. To keep the notation uncluttered, we omit the iteration index k. For τ^p, we have

$$\tau^p = \frac{1}{m}\left|\left|\boldsymbol{A}\right|\right|_F^2 \tau^x \overset{(a)}{=} \frac{1}{m} n \tau^x \overset{(b)}{=} \delta^{-1}\tau^x \tag{B.7}$$

where it is used for (a) that $\left|\left|\boldsymbol{A}\right|\right|_F^2 = n$ when the columns of $\boldsymbol{A}$ has unit ℓ_2-norm and for (b) it is used that $\delta = \frac{m}{n}$. For τ^s:

$$\tau^s = -\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial \hat{p}} g_{out}\left(\hat{p}_i, y_i, \tau^p\right) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau^p + \sigma^2} = \frac{1}{\tau^p + \sigma^2} = \frac{1}{\delta^{-1}\tau^x + \sigma^2}$$

Using the result for τ^s, we get for τ^r

$$\begin{aligned} & \frac{1}{\tau^r} = \frac{1}{n}\left|\left|\boldsymbol{A}\right|\right|_F^2 \tau^s = \frac{n}{n}\tau^s = \frac{1}{\tau^p + \sigma^2} \\ \iff \quad & \tau^r = \tau^p + \sigma^2 = \delta^{-1}\tau^x + \sigma^2 \end{aligned}$$

That is, using these assumptions we can express all the variance parameters in terms of τ^x.

Based on f_{in}, we can determine the input scalar function g_{in}:

$$\begin{aligned} g_{in}\left(\hat{r}, q, \tau^r\right) &\equiv \arg\max_x \left\{ f_{in}(x, q) - \frac{1}{2\tau^r}\left(\hat{r} - x\right)^2 \right\} \\ &= \arg\max_x \left\{ \ln\frac{\beta}{2} - \beta\left|x\right| - \frac{1}{2\tau^r}\left(\hat{r} - x\right)^2 \right\} \\ &= \arg\max_x \left\{ -\beta\left|x\right| - \frac{1}{2\tau^r}\left(\hat{r} - x\right)^2 \right\} \end{aligned}$$

In the above, we would like to the large $\beta \to \infty$ limit as done in the AMP derivation. But in order to do that we need to rewrite the above such that β is a factor on each of the two terms. Hence, we need to show that $\frac{1}{\tau^r} \propto \beta$. We have

$$\frac{1}{\tau^r} = \frac{1}{\tau^p + \sigma^2} = \frac{1}{\delta^{-1}\tau^x + \sigma^2} \overset{(a)}{=} \frac{1}{\delta^{-1}\tau^x + \frac{1}{\beta}} = \frac{\beta}{\frac{\beta}{\delta}\tau^x + 1}$$

where it is used for (a) that $\sigma^2 = \frac{1}{\beta}$. Therefore, we need $\tau^x \propto \beta^{-1}$. Using the expression for the variances parameter from above, we can rewrite the expression for τ^x as follows:

$$\begin{aligned}
\tau^x &= \frac{\tau^r}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right) \\
&= \frac{\tau^p + \sigma^2}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right) \\
&= \frac{\delta^{-1}\tau^x + \sigma^2}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right) \\
&= \frac{\delta^{-1}\tau^x}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right) + \frac{\sigma^2}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right) \\
\Longleftrightarrow \quad \tau^x\left(1 - \frac{\delta^{-1}}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right)\right) &= \frac{\sigma^2}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right) \\
\Longleftrightarrow \quad \tau^x &= \frac{\frac{\sigma^2}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right)}{1 - \frac{\delta^{-1}}{n}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right)}
\end{aligned}$$

Finally, we reduce and again use the fact that $\sigma^2 = \frac{1}{\beta}$:

$$\begin{aligned}
\tau^x &= \sigma^2\frac{\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right)}{n - \frac{1}{\delta}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right)} = \frac{1}{\beta}\frac{\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right)}{n - \frac{1}{\delta}\sum_{j=1}^{n}\frac{\partial}{\partial\hat{r}}g_{in}\left(\hat{r}_j, \tau^r\right)} \\
\Rightarrow \quad \tau^x &\propto \frac{1}{\beta}
\end{aligned}$$

as desired. Thus, for the specific choice of prior, the input $g_{in}(\cdot)$ becomes the soft thresholding function in the large β limit.

Consider now the expression for $\tau^r \hat{s}_i$:

$$\begin{aligned}
\tau^r \hat{s}_i &= \tau^r g_{out}\left(\hat{p}_i, y_i, \tau^p\right) = \tau^r \frac{y_i - \hat{p}_i}{\tau^r} = y_i - \hat{p}_i \\
&= y_i - \sum_j A_{ij}\hat{x}_j - \tau^p \hat{s}_i \\
&= y_i - \sum_j A_{ij}\hat{x}_j - \frac{1}{\delta}\tau^x \hat{s}_i \\
&= y_i - \sum_j A_{ij}\hat{x}_j - \frac{1}{\delta}\tau^r \hat{s}_i \frac{1}{n}\sum_{j=1}^{n} \frac{\partial}{\partial \hat{r}} g_{in}\left(\hat{r}_j, \tau^r\right) \\
&= y_i - \sum_j A_{ij}\hat{x}_j - \frac{1}{\delta}\tau^r \hat{s}_i \left\langle \frac{\partial}{\partial \hat{r}} g_{in}\left(\hat{r}_j, \tau^r\right) \right\rangle
\end{aligned}$$

Hence, we have that the quantity $\tau^r \boldsymbol{s}$ is equivalent to $\boldsymbol{z}$ in Donoho's AMP0. We now consider the expression for $\hat{r}_j$:

$$\hat{r}_k = \hat{x}_k + \tau^r \sum_{i=1}^{m} A_{ik}\hat{s}_i = \hat{x}_k + \sum_{i=1}^{m} A_{ik}\tau^r \hat{s}_i$$

Inserting this result into the expression for $\hat{x}_j$ yields:

$$\begin{aligned}
\hat{x}_j &= g_{in}\left(\hat{r}_j\ ,\ \tau^r\right) \\
&= g_{in}\left(\hat{x}_k + \sum_{i=1}^{m} A_{ik}\tau^r \hat{s}_i\ ;\ \tau^r\right)
\end{aligned}$$

Thus, this update equation has the same functional form as in Donoho's AMP0 given g_{in} is the soft thresholding function. Finally, we consider the expression for the threshold τ^r

$$\begin{aligned}
\tau^r &= \frac{1}{\beta} + \frac{1}{\delta}\tau^x \\
&= \frac{1}{\beta} + \frac{\tau^r(t)}{\delta}\frac{1}{n}\sum_{j=1}^{n} \frac{\partial}{\partial \hat{r}} g_{in}\left(\hat{r}_j, \tau^r\right) \\
&= \frac{1}{\beta} + \frac{\tau^r(t)}{\delta}\left\langle \frac{\partial}{\partial \hat{r}} g_{in}\left(\hat{r}_j, \tau^r\right) \right\rangle
\end{aligned}$$

Therefore, in the limit $\beta \to \infty$, τ^r has the same role as the threshold in AMP0.

Appendix C

MMV

C.1 Taylor Approximation for AMP-MMV

This appendix describes the Taylor approximation to one of the messages in the AMP-MMV algorithm. The objective here is to do a second order Taylor expansion of the $f(\theta)$ w.r.t θ around ϕ, where

$$\begin{aligned} f(\theta) &= -\ln g(\theta) = -\ln \hat{\mu}_{p(x_i^t \middle| \theta_i^t, s_i) \to \theta_i^t}(\theta_i^t) \\ &= \left(1 - \Omega\left(\overrightarrow{\pi}_i^t\right)\right) \mathcal{N}\left(\theta_i^t \middle| \frac{1}{\epsilon}\phi_i^t, {\frac{1}{\epsilon^2}}^{t}\right) + \Omega\left(\overrightarrow{\pi}_i^t\right) \mathcal{N}\left(\theta_i^t \middle| \phi^t, c^t\right), \end{aligned}$$

Thus,

$$\begin{aligned} f &\approx -f(\phi) - \frac{\partial f(\theta)}{\partial \theta}\Big|_{\theta=\phi}(\theta - \phi) - \frac{1}{2}\frac{\partial^2 f(\theta)}{\partial \theta^2}\Big|_{\theta=\phi}(\theta-\phi)^2 \\ &= K - \frac{1}{2}\frac{\partial^2 f(\theta)}{\partial \theta^2}\Big|_{\theta=\phi}\theta^2 + \left(\phi\frac{\partial^2 f(\theta)}{\partial \theta^2}\Big|_{\theta=\phi} - \frac{\partial f(\theta)}{\partial \theta}\Big|_{\theta=\phi}\right)\theta \end{aligned} \tag{C.1}$$

A second Taylor approximation of the logarithm of a function can be interpreted a Gaussian approximation to the function. Denote this Gaussian by $\mathcal{N}\left(\theta \middle| \xi, \psi\right)$:

$$\ln \mathcal{N}\left(\theta \middle| \overrightarrow{\xi}, \overrightarrow{\psi}\right) = K - \frac{\theta^2}{2\overrightarrow{\psi}} - \frac{\overrightarrow{\xi}^2}{2\overrightarrow{\psi}} + \frac{\theta\overrightarrow{\xi}}{\psi} \tag{C.2}$$

By comparing the two expression, we make the following identifications:

$$\frac{1}{\psi} = \frac{\partial^2 f(\theta)}{\partial \theta^2}\Big|_{\theta=\phi} \qquad\qquad \xi = \phi - \psi\frac{\partial f(\theta)}{\partial \theta}\Big|_{\theta=\phi} \tag{C.3}$$

So task is really to determine the first and second order partial derivatives. The first partial derivative of f is obtained as:

$$\frac{\partial}{\partial \theta} f(\theta) = \frac{g'(\theta)}{g(\theta)} \tag{C.4}$$

and the second

$$\frac{\partial^2}{\partial \theta^2} f(\theta) = \frac{g''(\theta)g(\theta) - g'(\theta)^2}{g(\theta)^2} \tag{C.5}$$

C.2 EM Update Equations for the AMP-MMV model

The purpose of this appendix is to describe the derivation of the EM update rules for the MMV-AMP model described in section 3.1.

Learning the Noise Variance

The approach is the same as for the noise variance for the BG-AMP model described in section 2.4. Therefore, similar to eq. (2.234), we get

$$\frac{\partial}{\partial\sigma^2}\mathcal{L}(\boldsymbol{Y}|\boldsymbol{q}) = \frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\ln p(\boldsymbol{X},\boldsymbol{Y}|\boldsymbol{q})\right]$$

Inserting the joint density:

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}\mathcal{L}(\boldsymbol{Y}|\boldsymbol{q}) &= \frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\ln\left(\prod_{t=1}^{T}\left[\prod_{a=1}^{m}p(y_a^t|\boldsymbol{x}^t)\prod_{i=1}^{n}p(x_i^t|\theta_i^t,s_i)p(\theta_i^t|\theta_i^{t-1})\right]\prod_{i=1}^{n}p(s_i)\right)\right]\\ &= \frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\sum_{t=1}^{T}\left(\ln p(\boldsymbol{y}^t|\boldsymbol{x}^t)+\sum_{i=1}^{n}\ln p(x_i^t|\theta_i^t,s_i)+\sum_{i=1}^{n}\ln p(\theta_i^t|\theta_i^{t-1})\right)+\ln\prod_{i=1}^{n}p(s_i)\right]\end{aligned} \tag{C.6}$$

The only term depending on the noise variance σ^2 is $\ln p(\boldsymbol{y}^t|\boldsymbol{x}^t)$ and therefore:

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}\mathcal{L}(\boldsymbol{Y}|\boldsymbol{q}) &= \frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\sum_{t=1}^{T}\ln p(\boldsymbol{y}^t|\boldsymbol{x}^t\right]\\ &= \frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{n}\ln p(\boldsymbol{y}_a^t|\boldsymbol{x}^t)\right]\\ &= \sum_{t=1}^{T}\sum_{i=1}^{n}\frac{\partial}{\partial\sigma^2}\mathbb{E}\left[\ln p(\boldsymbol{y}_a^t|\boldsymbol{x}^t)\right]\\ &= \sum_{t=1}^{T}\sum_{i=1}^{n}\frac{\partial}{\partial\sigma^2}\int p(z_a^t|y_a^t)\ln p(\boldsymbol{y}_a^t|\boldsymbol{x}^t)\,\mathrm{d}z_a^t\end{aligned}$$

Exchanging the order of the partial derivative and integration:

$$\begin{aligned}\frac{\partial}{\partial \sigma^2}\mathcal{L}(\boldsymbol{Y}|\boldsymbol{q}) &= \sum_{t=1}^{T}\sum_{i=1}^{n}\int p(z_a^t|y_a^t)\frac{\partial}{\partial \sigma^2}\ln p(\boldsymbol{y}_a^t|\boldsymbol{x}^t)\,\mathrm{d}z_a^t \\ &= \sum_{t=1}^{T}\sum_{i=1}^{n}\int p(z_a^t|y_a^t)\left(\frac{1}{2}\left[\frac{\left(y_a^t - z_a^t\right)^2}{\left(\sigma^2\right)^2} - \frac{1}{\sigma^2}\right]\right)\mathrm{d}z_a^t\end{aligned}$$

The integration is now straightforward and yields:

$$\frac{\partial}{\partial \sigma^2}\mathcal{L}(\boldsymbol{Y}|\boldsymbol{q}) = \frac{1}{2}\frac{1}{\left(\sigma^2\right)^2}\sum_{t=1}^{T}\sum_{i=1}^{n}\left[\left(y_a^t - z_a^t\right)^2 + \tau_a^z\right] - \frac{1}{2}\frac{1}{\sigma^2}\sum_{t=1}^{T}\sum_{i=1}^{n}1 = 0 \tag{C.7}$$

Solving for σ^2 now give the update rule:

$$\sigma_{new}^2 = \frac{1}{Tm}\sum_{t=1}^{T}\sum_{i=1}^{n}\left[(y_a^t - z_a^t)^2 + \tau_a^z\right] \tag{C.8}$$

Learning the Sparsity λ

For the E-step, we first obtain the posterior distribution over s_i given the current model. The is readily obtained by computing the product of the incoming messages at node s_i in the factor graph shown in figure 3.2:

$$\begin{aligned}p(s_i|\boldsymbol{Y}) &= \mu_{p(s_i)\rightarrow s_i}(s_i)\prod_{t=1}^{T}\mu_{p(x_i^t|\theta_i^t, s_i)\rightarrow s_i}(s_i) \\ &= \mathrm{Ber}(s_i|\lambda)\prod_{t=1}^{T}\mathrm{Ber}(s_i|\overleftarrow{\pi}_i^t) \\ &= \left[(1-\lambda)\prod_{t=1}^{T}\left(1-\overleftarrow{\pi}_i^t\right)\right](1-s_i) + \left[\lambda\prod_{t=1}^{T}\overleftarrow{\pi}_i^t\right]s_i \\ &= \left(1-\beta_i^t\right)(1-s_i) + \beta_i^t s_i\end{aligned} \tag{C.9}$$

where

$$\beta_i = \frac{\lambda\prod_{t=1}^{T}\overleftarrow{\pi}_i^t}{(1-\lambda)\prod_{t=1}^{T}(1-\overleftarrow{\pi}_i^t) + \lambda\prod_{t=1}^{T}\overleftarrow{\pi}_i^t} \tag{C.10}$$

Taking the partial derivative of the bound function w.r.t. λ:

$$\begin{aligned}\frac{\partial}{\partial\lambda}\mathcal{L}\left(\boldsymbol{Y};\boldsymbol{q}\right) &= \frac{\partial}{\partial\lambda}\mathbb{E}\left[\ln\prod_{t=1}^{T} p\left(\boldsymbol{y}^t\middle|\boldsymbol{x}^t\right)p(\boldsymbol{x}^t|\boldsymbol{q})\right]\\ &= \frac{\partial}{\partial\lambda}\mathbb{E}\left[\sum_{t=1}^{T}\left[\ln p(\boldsymbol{y}^t|\boldsymbol{x}^t)+\sum_{i=1}^{n}\ln p(x_i^t|\theta_i^t,s_i)+\sum_{i=1}^{n}\ln p(\theta_i^t|\theta_i^{t-1})\right]+\ln\prod_{i=1}^{n}p(s_i)\right]\end{aligned}$$

The $p(s_i)$-terms are the only terms depending on λ, therefore

$$\begin{aligned}\frac{\partial}{\partial\lambda}\mathcal{L}\left(\boldsymbol{Y};\boldsymbol{q}\right) &= \sum_{i=1}^{n}\frac{\partial}{\partial\lambda}\mathbb{E}\left[\ln p(s_i)\right]\\ &= \sum_{i=1}^{n}\mathbb{E}\left[\frac{\partial}{\partial\lambda}\ln p(s_i)\right]\\ &= \sum_{i=1}^{n}\sum_{s_i}\left[(1-\beta_i)(1-s_i)+\beta_i s_i\right]\frac{\partial}{\partial\lambda}\ln p(s_i)\end{aligned}$$

The partial derivative evaluates to

$$\begin{aligned}\frac{\partial}{\partial\lambda}\ln p(s_i) &= \frac{1}{p(s_i)}\left(2s-1\right)\\ &= \begin{cases}\frac{-1}{1-\lambda} & \text{if } s=0\\ \frac{1}{\lambda} & \text{if } s=1\end{cases}\end{aligned} \tag{C.11}$$

Inserting this into eq. (C.11) and plugging in the expression for the posterior distribution yields:

$$\frac{\partial}{\partial\lambda}\mathcal{L}\left(\boldsymbol{Y};\boldsymbol{q}\right) = \sum_{i=1}^{n}\left[(1-\beta_i^t)\frac{-1}{1-\lambda}+\beta_i^t\frac{1}{\lambda}\right] = 0 \tag{C.12}$$

Solving for λ yields the update equation:

$$\lambda^{new} = \frac{1}{n}\sum_{i=1}^{n}\beta_i^t \tag{C.13}$$

and substituting in the expression for β:

$$\lambda^{new} = \frac{1}{n}\sum_{i=1}^{n}\frac{\lambda\prod_{t=1}^{T}\overleftarrow{\pi}_i^t}{(1-\lambda)\prod_{t=1}^{T}(1-\overleftarrow{\pi}_i^t)+\lambda\prod_{t=1}^{T}\overleftarrow{\pi}_i^t} \tag{C.14}$$

Learning the Mean ζ

Formally, we need the pair-wise joint $p\left(\theta_i^t, \theta_i^{t-1} \middle| \boldsymbol{Y}\right)$, but later when we have to compute the M-step, θ_i^t and θ_i^{t-1} appear only in linear combinations, so the can settle for the marginals $p\left(\theta_i^t \middle| \boldsymbol{Y}\right)$. Therefore, for the E-step:

$$\begin{aligned} p(\theta_i^t|\boldsymbol{Y}) &= \mu_{p(\theta_i^t|\theta_i^{t-1})\to\theta_i^t}\left(\theta_i^t\right) \mu_{p(\theta_i^{t+1}|\theta_i^t)\to\theta_i^t}\left(\theta_i^t\right) \mu_{p(x_i^t|\theta_i^t, s_i)\to\theta_i^t}\left(\theta_i^t\right) \\ &= \mathcal{N}\left(\theta_i^t \middle| \overleftarrow{\eta}_i^t, \overleftarrow{\kappa}_i^t\right) \mathcal{N}\left(\theta_i^t \middle| \overrightarrow{\eta}_i^t, \overrightarrow{\kappa}_i^t\right) \mathcal{N}\left(\theta_i^t \middle| \overleftarrow{\xi}_i^t, \overleftarrow{\psi}_i^t\right) \\ &\propto \mathcal{N}\left(\theta_i^t \middle| \hat{\theta}_i^t, \tilde{\theta}_i^t\right) \end{aligned} \tag{C.15}$$

where $\hat{\theta}_i^t$ and $\tilde{\theta}_i^t$ are obtained by 2 applications of the Gaussian multiplication rule, respectively:

$$\hat{\theta}_i^t = \tilde{\theta}_i^t \left(\frac{\overrightarrow{\eta}_i^t}{\overrightarrow{\kappa}_i^t} + \frac{\overleftarrow{\eta}_i^t}{\overleftarrow{\kappa}_i^t} + \frac{\overleftarrow{\xi}_i^t}{\overleftarrow{\psi}_i^t} \right) \qquad \tilde{\theta}_i^t = \left(\frac{1}{\overrightarrow{\kappa}_i^t} + \frac{1}{\overleftarrow{\kappa}_i^t} + \frac{1}{\overleftarrow{\psi}_i^t} \right)^{-1} \tag{C.16}$$

For the M-step:

$$\begin{aligned} \frac{\partial}{\partial\zeta} L(\boldsymbol{y}|\boldsymbol{q}) &= \frac{\partial}{\partial\zeta}\mathbb{E}\left[p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{q})\right] \\ &= \frac{\partial}{\partial\zeta}\mathbb{E}\left[\ln \prod_{t=1}^{T} \left[p(\boldsymbol{y}^t|\boldsymbol{x}^t) \prod_{i=1}^{n} p(x_i^t|\theta_i^t, s_i) p(\theta_i^t|\theta_i^{t-1}) \right] \prod_{i=1}^{n} p(s_i) \right] \\ &= C \sum_{t=1}^{T}\sum_{i=1}^{n} \frac{\partial}{\partial\zeta}\mathbb{E}\left[\ln p(\theta_i^t|\theta_i^{t-1})\right] \\ &= C \sum_{t=1}^{T}\sum_{i=1}^{n} \int p(\theta_i^t, \theta_i^{t-1}|\boldsymbol{Y}) \frac{\partial}{\partial\zeta} \ln p(\theta_i^t|\theta_i^{t-1})\, \mathrm{d}\theta_i^t \end{aligned} \tag{C.17}$$

where $p(\theta_i^t|\theta_i^{t-1})$ is given by eq. (3.6) and eq. (3.7) for $t > 1$ and $t = 1$, respectively. Computing the partial derivatives yields:

$$\begin{aligned} \frac{\partial}{\partial\zeta} L(\boldsymbol{y}|\boldsymbol{q}) = C \sum_{i=1}^{n} \int p(\theta_i^1, \theta_i^{t-1}|\boldsymbol{Y}) \frac{\theta_i^1 - \zeta}{\sigma_c^2}\, \mathrm{d}\theta_i^1\, \mathrm{d}\theta_i^{t-1} \\ + C \sum_{t=1}^{T}\sum_{i=1}^{n} \int p(\theta_i^t, \theta_i^{t-1}|\boldsymbol{Y}) \frac{1}{\alpha\rho} \left[\theta_i^t - (1-\alpha)\left(\theta_i^{t-1} - \zeta\right) - \zeta\right] \mathrm{d}\theta_i^t\, \mathrm{d}\theta_i^{t-1} \\ = C \frac{1}{\sigma_c^2} \sum_{i=1}^{n} \left[\hat{\theta}_i^1 - \zeta\right] + C \sum_{t=2}^{T}\sum_{i=1}^{n} \frac{1}{\alpha\rho}\left[\hat{\theta}_i^t - (1-\alpha)\hat{\theta}_i^{t-1} - \alpha\zeta\right] \end{aligned}$$

Equating the above equation to zero and solving for ζ then yields the update equation:

$$\zeta^{new} = \left(\frac{(T-1)n}{\rho} + \frac{n}{\sigma_c^2}\right)^{-1} \left(\frac{1}{\sigma_c^2}\sum_{i=1}^{n}\hat{\theta}_i^1 + \frac{1}{\alpha\rho}\sum_{t=2}^{T}\sum_{i=1}^{n}\left[\hat{\theta}_i^t - (1-\alpha)\hat{\theta}_i^{t-1}\right]\right) \tag{C.18}$$

Learning the Variance ρ

To derive the update rule for the variance parameter ρ, we do need the pairwise posterior $p\left(\theta_i^t, \theta_i^{t-1} \middle| \boldsymbol{Y}\right)$, which is readily obtained:

$$\begin{aligned} p\left(\theta_i^t, \theta_i^{t-1} \middle| \boldsymbol{Y}\right) &= p\left(\theta_i^t \middle| \theta_i^{t-1}\right) \cdot \mu_{\theta_i^t \to p\left(\theta_i^t \middle| \theta_i^{t-1}\right)}\left(\theta_i^t\right) \cdot \mu_{\theta_i^{t-1} \to p\left(\theta_i^t \middle| \theta_i^{t-1}\right)}\left(\theta_i^{t-1}\right) \\ &= p\left(\theta_i^t \middle| \theta_i^{t-1}\right) \cdot \mathcal{N}\left(\theta_i^{t-1} \middle| \overrightarrow{\eta}_i^{t-1}, \overrightarrow{\kappa}_i^{t-1}\right) \mathcal{N}\left(\theta_i^{t-1} \middle| \overleftarrow{\xi}_i^{t-1}, \overleftarrow{\psi}_i^{t-1}\right) \cdot \mathcal{N}\left(\theta_i^t \middle| \overleftarrow{\eta}_i^t, \overleftarrow{\kappa}_i^t\right) \end{aligned} \tag{C.19}$$

Similar to eq. (C.17), we get the M-step:

$$\frac{\partial}{\partial\rho}L(\boldsymbol{y}|\boldsymbol{q}) = C\sum_{t=1}^{T}\sum_{i=1}^{n}\int p(\theta_i^t, \theta_i^{t-1}|\boldsymbol{Y})\frac{\partial}{\partial\rho}\ln p(\theta_i^t|\theta_i^{t-1})\,\mathrm{d}\theta_i^t\theta_i^{t-1}$$

The partial derivative w.r.t ρ for $t > 1$ is

$$\frac{\partial}{\partial\rho}\ln p(\theta_i^t|\theta_i^{t-1}) = \frac{\left(\theta_i^t - (1-\alpha)\left(\theta_i^{t-1} - \zeta\right) - \zeta\right)^2 - \alpha^2\rho}{2\alpha^4\rho^2} \tag{C.20}$$

and for $t = 1$:

$$\frac{\partial}{\partial\rho}\ln p(\theta_i^t|\theta_i^{t-1}) = 0 \tag{C.21}$$

since the conditional density $p(\theta_i^t|\theta_i^{t-1})$ is independent of ρ. Inserting the partial derivatives, equating to zero, and solving for ρ yields:

$$\begin{aligned} \rho = \frac{1}{\alpha^2(T-1)N}\sum_{t=2}^{T}\sum_{i=1}^{n}[&\mathbb{E}\left[\left(\theta_i^t\right)^2|\boldsymbol{Y}\right] - 2\zeta\alpha\hat{\theta}^t + (1-\alpha)^2\left(\mathbb{E}\left[\left(\theta_i^{t-1}\right)^2|\boldsymbol{Y}\right]\right) + \zeta^2\alpha^2 \\ &+ 2\zeta\alpha(1-\alpha)\hat{\theta}_i^{t-1} - 2(1-\alpha)\mathbb{E}\left(\theta_i^t\theta_i^{t-1}|\boldsymbol{Y}\right)] \end{aligned} \tag{C.22}$$

which is equivalent to:

$$\begin{aligned} \rho = \frac{1}{\alpha^2(T-1)N}\sum_{t=2}^{T}\sum_{i=1}^{n}[&\tilde{\theta}_i^t + \left(\hat{\theta}_i^t\right)^2 - 2\zeta\alpha\hat{\theta}^t + (1-\alpha)^2\left(\tilde{\theta}_i^{t-1} + \hat{\theta}_i^{t-1}\right) + \zeta^2\alpha^2 \\ &+ 2\zeta\alpha(1-\alpha)\hat{\theta}_i^{t-1} - 2(1-\alpha)\mathbb{E}\left(\theta_i^t\theta_i^{t-1}|\boldsymbol{Y}\right)] \end{aligned} \tag{C.23}$$

Learning the Correlation Parameter α

Similar to eq. (C.17), we get

$$\frac{\partial}{\partial \alpha} L(\boldsymbol{y}|\boldsymbol{q}) = C \sum_{t=1}^{T} \sum_{i=1}^{n} \int p(\theta_i^t, \theta_i^{t-1} | \boldsymbol{Y}) \frac{\partial}{\partial \alpha} \ln p(\theta_i^t | \theta_i^{t-1}) \, \mathrm{d}\theta_i^t \, \mathrm{d}\theta_i^{t-1}$$

For $t > 1$, the partial derivative is:

$$\begin{aligned} \frac{\partial}{\partial \alpha} \ln p(\theta_i^t | \theta_i^{t-1}) = -\frac{1}{\alpha^3 \rho} \big(& \alpha^2 \rho - \alpha \theta_i^t \theta_i^{t-1} + \alpha \zeta \theta_i^t + \alpha \left(\theta_i^{t-1}\right)^2 - \alpha \zeta \theta_i^{t-1} - \left(\theta_i^t\right)^2 \\ & - 2\theta_i^t \theta_i^{t-1} - \left(\theta_i^{t-1}\right)^2 \big) \end{aligned} \tag{C.24}$$

and for $t = 1$, the partial derivative is 0. Plugging the derivative back in, equating to zero and solving for α yields the update equation:

$$\alpha^{new} = \frac{b - \sqrt{b^2 + 4N(T-1)c}}{2N(T-1)} \tag{C.25}$$

with

$$b = \frac{1}{\rho} \sum_{t=2}^{T} \sum_{i=1}^{n} \left(\mathbb{E}\left[\theta_i^t \theta_i^{t-1} | \boldsymbol{Y}\right] - \zeta \left(\mathbb{E}\left[\theta_i^t | \boldsymbol{Y}\right] - \mathbb{E}\left[\theta_i^{t-1} | \boldsymbol{Y}\right] \right) - \mathbb{E}\left[\left(\theta_i^{t-1}\right)^2 | \boldsymbol{Y} \right] \right) \tag{C.26}$$

and

$$c = \frac{1}{\rho} \sum_{t=2}^{T} \sum_{i=1}^{n} \left(\mathbb{E}\left[\left(\theta_i^t\right)^2 | \boldsymbol{Y} \right] - 2\mathbb{E}\left[\theta_i^t \theta_i^{t-1} | \boldsymbol{Y}\right] + \mathbb{E}\left[\left(\theta_i^{t-1}\right)^2 | \boldsymbol{Y} \right] \right) \tag{C.27}$$

C.3 EM Update Equation for AMP-DCS

The purpose of this appendix is to describe to EM update rule specific to the AMP-DCS model. For more details of the general approach, cf. section 2.4 and appendix C.2.

Learning the Sparsity

It is seen from figure 3.10, that the posterior of s_i^t given the measurements $\boldsymbol{Y}$ is given by:

$$p(s_i^t|\boldsymbol{Y}) = \mu_{p(s_i^t|s_i^{t-1})\to s_i^t}(s_i^t) \cdot \mu_{p(s_i^{t+1}|s_i^t)\to s_i^t}(s_i^t) \cdot \mu_{p(x_i^t|\theta_i^t, s_i^t)\to s_i^t}(s_i^t)$$

Inserting the three messages and reducing leads to a posterior of the form:

$$p(s_i^t|\boldsymbol{Y}) \propto \left[(1-\overrightarrow{\lambda}_i^t)(1-\overleftarrow{\lambda}_i^t)(1-\overleftarrow{\pi}_i^t)\right](1-s_i^t) + \left[\overrightarrow{\lambda}_i^t\overleftarrow{\lambda}_i^t\overleftarrow{\pi}_i^t\right] s_i^t \tag{C.28}$$

Taking the partial derivative of the bound function w.r.t. λ:

$$\begin{aligned}\frac{\partial}{\partial\lambda}\mathcal{L}(\boldsymbol{Y};\boldsymbol{q}) &= \frac{\partial}{\partial\lambda}\mathbb{E}\left[\ln\prod_{t=1}^{T} p\left(\boldsymbol{y}^t|\boldsymbol{x}^t\right) p(\boldsymbol{x}^t|\boldsymbol{q})\right]\\ &= \frac{\partial}{\partial\lambda}\mathbb{E}\left[\sum_{t=1}^{T}\left[\ln p(\boldsymbol{y}^t|\boldsymbol{x}^t) + \sum_{i=1}^{n}\ln p(x_i^t|\theta_i^t, s_i) + \sum_{i=1}^{n}\ln p(\theta_i^t|\theta_i^{t-1}) + \sum_{i=1}^{n}\ln p(s_i^t|s_i^{t-1})\right]\right]\end{aligned}$$

The only terms depending on λ is the prior on the support variables for $t=1$, therefore we get

$$\frac{\partial}{\partial\lambda}\mathcal{L}(\boldsymbol{Y};\boldsymbol{q}) = \frac{\partial}{\partial\lambda}\mathbb{E}\left[\sum_{i=1}^{n}\ln p(s_i^1|s_i^0)\right]$$

where $p(s_i^1|s_i^0) = p(s_i) = \text{Ber}(\lambda)$. Following the approach the same approach for the AMP-MMV model (see appendix C.2), we get the update equation:

$$\lambda^{new} = \frac{1}{n}\sum_{i=1}^{n}\frac{\overrightarrow{\lambda}_i^1\overleftarrow{\lambda}_i^1\overleftarrow{\pi}_i^1}{(1-\overrightarrow{\lambda}_i^1)(1-\overleftarrow{\lambda}_i^1)(1-\overleftarrow{\pi}_i^1) + \overrightarrow{\lambda}_i^1\overleftarrow{\lambda}_i^1\overleftarrow{\pi}_i^1} \tag{C.29}$$

Learning the transitional probability

The relevant posterior distribution is obtained from the sum-product algorithm:
E-step:

$$\begin{aligned}p(s_i^t, s_i^{t-1}\big|\,\boldsymbol{Y}) &= p(s_i^t|s_i^{t-1}) \cdot \mu_{s_i^{t-1}\to p(s_i^t|s_i^{t-1})}(s_i^{t-1}) \cdot \mu_{s_i^t \to p(s_i^t|s_i^{t-1})}(s_i^t) \\ &= p(s_i^t|s_i^{t-1})\text{Ber}\left(s_i^{t-1}\big|\,\overrightarrow{\lambda}_i^{t-1}\right)\text{Ber}\left(s_i^{t-1}\big|\overleftarrow{\pi}_i^{t-1}\right)\text{Ber}\left(s_i^t\big|\,\overrightarrow{\lambda}_i^t\right)\text{Ber}\left(s_i^t\big|\overleftarrow{\pi}_i^t\right)\end{aligned} \quad \text{(C.30)}$$

Taking the partial derivative of the bound function w.r.t. p_{10}:

$$\begin{aligned}\frac{\partial}{\partial p_{10}}\mathcal{L}\left(\boldsymbol{Y};\boldsymbol{q}\right) &= \frac{\partial}{\partial p_{10}}\mathbb{E}\left[\sum_{t=2}^{T}\sum_{i=1}^{n}\ln p(s_i^t|s_i^{t-1})\right] \\ &= \mathbb{E}\left[\sum_{t=2}^{T}\sum_{i=1}^{n}\frac{\partial}{\partial p_{10}}\ln p(s_i^t|s_i^{t-1})\right]\end{aligned} \quad \text{(C.31)}$$

where it is used that the only terms, which depend on p_{10} is $\ln p(s_i^t|s_i^{t-1})$ for $t > 1$. In order to compute the derivative, we utilize that $p(s_i^t|s_i^{t-1})$ can be written as:

$$p(s_i^t|s_i^{t-1}) = (1-s_i^{t-1})(1-s_i^t)\cdot p_{00} + (1-s_i^{t-1})s_i^t \cdot p_{01} + s_i^{t-1}(1-s_i^t)\cdot p_{10} + s_i^{t-1}s_i^t \cdot p_{11}$$

Using the expression above and the fact that $p_{10} + p_{11} = 1$, the derivative is easily evaluated as:

$$\begin{aligned}\frac{\partial}{\partial p_{10}}\ln p(s_i^t|s_i^{t-1}) &= \frac{1}{p(s_i^t|s_i^{t-1})}\frac{\partial}{\partial p_{10}}p(s_i^t|s_i^{t-1}) \\ &= \frac{s_i^{t-1}(1-s_i^t) - s_i^{t-1}s_i^t}{p(s_i^t|s_i^{t-1})}\end{aligned}$$

From the numerator, it is seen that the term $s_i^{t-1}(1-s_i^t)$ is only non-zero if $s^{t-1} = 1$ and $s_i^t = 0$ and the term $s_i^{t-1}s_i^t$ is only non-zero if both variable are equal to 1. Therefore, we can write:

$$\frac{\partial}{\partial p_{10}}\ln p(s_i^t|s_i^{t-1}) = \frac{s_i^{t-1}(1-s_i^t)}{p_{10}} - \frac{s_i^{t-1}s_i^t}{1-p_{10}} \quad \text{(C.32)}$$

Plugging this back into eq. C.31 and rearranging:

$$\begin{aligned}\frac{\partial}{\partial p_{10}}\mathcal{L}\left(\boldsymbol{Y};\boldsymbol{q}\right) &= \sum_{t=2}^{T}\sum_{i=1}^{n}\left(\frac{1}{p_{10}}\mathbb{E}\left[s_i^{t-1}(1-s_i^t)\right] - \frac{1}{1-p_{10}}\mathbb{E}\left[s_i^{t-1}s_i^t\right]\right) \\ &= \sum_{t=2}^{T}\sum_{i=1}^{n}\left(\frac{1}{p_{10}}\mathbb{E}\left[s_i^{t-1}\right] - \frac{1}{p_{10}}\mathbb{E}\left[s_i^{t-1}s_i^t\right] - \frac{1}{1-p_{10}}\mathbb{E}\left[s_i^{t-1}s_i^t\right]\right)\end{aligned}$$

Finally, putting the expression equal to zero and solving for p_{10} yields the update equation:

$$p_{10}^{\text{new}} = \frac{\sum_{t=2}^{T} \sum_{i=1}^{n} \mathbb{E}\left[s_i^{t-1}\right] - \sum_{t=2}^{T} \sum_{i=1}^{n} \mathbb{E}\left[s_i^{t-1} s_i^{t}\right]}{\sum_{t=2}^{T} \sum_{i=1}^{n} \mathbb{E}\left[s_i^{t-1}\right]}, \tag{C.33}$$

where the moments are obtained from the distribution in eq. C.30.

Appendix D

EEG Source Localization

The purpose of this appendix is motivate the study of the linear inverse problems by giving a brief introduction to source localization based on *electroencephalography* (EEG). Furthermore, we will argue that the underlying problem indeed is linear. This introduction is by no means intended to be a thorough introduction to field of EEG, but instead the interested reader are referred to [BML01, NS06, Mic09] for details and references.

A scalp EEG recording is a multivariate time series consisting of measurements of electromagnetic fields generated by the human brain. The electric potentials are measured between pairs of electrodes attached directly to the skin on the scalp of a human [NS06]. EEG recordings are non-invasive, cheap and portable, which makes EEG an attractive tool and compared to other non-invasive methods, like functional magnetic resonance imagery (fMRI) and positron emission tomography (PET) [BML01]. Moreover, EEG has significantly higher temporal resolution, but lower spatial resolution than fMRI and PET [BML01].

For many years, EEG has served as an important clinical tool for neurological diseases, e.g. epilepsies [NS06]. A more recent application of EEG is the problem of *source localization*. The goal is here to locate and identify the brain regions whose neural activations generated the electric potentials measured at the scalp. Thus, in this application the electric scalp potentials are not of direct interest themselves, but rather the *sources* that generated them. For this reason, this is known as the *source localization* problem. Locating and identifying the EEG sources are important for both clinical applications [PHC08] and for increasing the basic understanding of how the human brain works. In the framework of inverse problems, the underlying sources constitute the hidden state and the scalp potentials constitute the measurements. Thus, given knowledge of the sources, the forward problem amounts to compute the resulting electric potentials, while the inverse problem amounts to estimating the sources based on the measured potentials. This is illustrated in figure D.1.

In order to solve the source localization problem, it is necessary to model the relationship between the measured potentials and the underlying sources of interest. To achieve this, the desired current distribution of the brain can be approximated using the *equivalent current dipole* model, which assumes that the current distribution can be represented using a discrete set of current dipoles, i.e. a current source and a current sink [BML01]. These point sources give rise to electromagnetic fields since the head (brain tissue, skull, scalp etc.) behaves a volume conductor. Using an appropriate model for the head, Maxwell's equations governs the relationship between the orientations and magnitudes of the dipoles and the measured potentials at the surface of the scalp. In most studies the frequency content of interest is below 100Hz and consequently, the governing equations can be significantly simplified using the quasi-static approximation of Maxwell's equations [BML01].

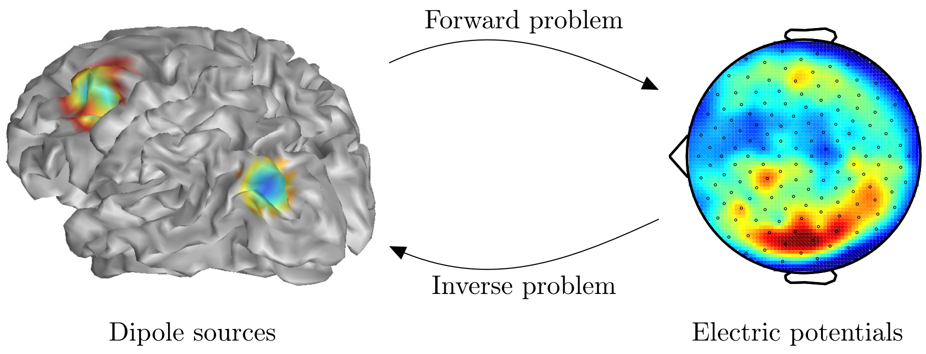


Figure D.1: Illustration of the forward-inverse relationship for EEG source localization. Inspired from [Sta08]

The task of computing the resulting scalp potentials given knowledge of point sources is the *EEG forward problem*. Solving the forward problem includes obtaining a suitable *head model*, which describes the geometry and conductivity of the human head. Head models vary from simple nested concentric spherical shells with homogeneous conductivity to realistic head models extracted from magnetic resonance or computer tomography imagery [HVG+07]. For sufficiently simple models, analytically solutions exist to the forward problem, but for more complex head models numerical methods are required [MLL99].

Due to the quasi-static approximation, the relationship between the magnitude of a dipole source x and a scalp potential y is given by

$$y(\boldsymbol{r}) = a(\boldsymbol{r}, \boldsymbol{p}, \theta)x \tag{D.1}$$

where $\boldsymbol{r}$ is the measurement position, $\boldsymbol{p}$ is the dipole position, θ is the dipole orientation, x is the dipole magnitude and $a(\boldsymbol{r}, \boldsymbol{p}, \theta)$ is obtained from the solution of Maxwell's equations [BML01]. Note that $y(\boldsymbol{r})$ is only linear w.r.t. the dipole magnitude x. Using linear superposition, the relationship between multiple sources $\{x_i\}_{i=1}^n$ and multiple scalp potentials $\{y_a\}_{a=1}^m$ can be expressed as:

$$\underbrace{\begin{bmatrix} y(\boldsymbol{r}_1) \\ y(\boldsymbol{r}_2) \\ \vdots \\ y(\boldsymbol{r}_m) \end{bmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{bmatrix} a(\boldsymbol{r}_1, \boldsymbol{p}_1, \theta_1) & \dots & a(\boldsymbol{r}_1, \boldsymbol{p}_n, \theta_n) \\ \vdots & \ddots & \vdots \\ a(\boldsymbol{r}_m, \boldsymbol{p}_1, \theta_1) & \dots & a(\boldsymbol{r}_m, \boldsymbol{p}_n, \theta_n) \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\boldsymbol{x}}$$

Thus, the underlying problem is indeed linear and therefore the source localization problem belongs to the class of linear inverse problems.

The number of electrodes can be as low 1 and as many as 256, while the number of sources be as high 300000 [BLP+12], but typically the number of measurements is in order of tens and the number of sources is in the order of thousands [NS06]. Furthermore, adjacent electrodes often fail to provide independent information due to strong correlations [GYO+04]. Therefore, the source localization problem is ill-posed by nature.

Imposing sparsity on the estimated sources is convenient from a mathematical point of view because of the ill-posed nature of the problem. But recent evidence suggests that sparsity in fact is a reasonable assumption [OPD+12], e.g. in [GKH12] it is argued that during a given cognitive task only a few region a activated simultaneously. However, in [NS06], it is stated that even if the true underlying source are not sparse, it would be natural to aim to retrieve the most active sources. Thus, the EEG source localization problem is well suited for the sparsity framework. Furthermore, the dynamic nature of the EEG sources [NS06, AM12] makes the problem suitable for MMV models with dynamic support

Appendix E

MLSP13 Paper on The Variational Garrote

LEARNING THE SOLUTION SPARSITY OF AN ILL-POSED LINEAR INVERSE PROBLEM WITH THE VARIATIONAL GARROTE

Michael Riis Andersen, Sofie Therese Hansen, Lars Kai Hansen

Department of Applied Mathematics and Computer Science
Technical University of Denmark B303
Kongens Lyngby, Denmark

ABSTRACT

The Variational Garrote is a promising new approach for sparse solutions of ill-posed linear inverse problems (Kappen and Gomez, 2012). We reformulate the prior of the Variational Garrote to follow a simple Binomial law and assign a Beta hyper-prior on the parameter. With the new prior the Variational Garrote, we show, has a wide range of parameter values for which it at the same time provides low test error and high retrieval of the true feature locations. Furthermore, the new form of the prior and associated hyper-prior leads to a simple update rule in a Bayesian variational inference scheme for its hyperparameter. As a second contribution we provide evidence that the new procedure can improve on cross-validation of the parameters and we find that the new formulation of the prior outperforms the original formulation when both are cross-validated to determine hyperparameters.

Index Terms— Ill-posed inverse problem, linear regression, Variational Garrote

1. INTRODUCTION

Finding robust and accurate solutions to ill-posed linear inverse problems is of both theoretical and practical importance. A great variety of methods have been proposed. A large and useful class of methods is based on sparse representations assuming that only a small subset of the underdetermined variables are non-zero or alternatively that a small number of linear combinations of these variables are non-zero. Finding such subsets leads to hard combinatorial optimization in general, therefore a number of approximations have been proposed, e.g., convex relaxations such as the widely used LASSO [1]. Convex relaxations, however, suffer from well-known issues with consistency and lack of precision [2]. Automatic Relevance Determination (ARD) is a popular alternative, aiming directly at the combinatorial problem [3]. The basic idea is to use adaptive regularization so that weak features experience large regularization, hence are pruned, see e.g. [4]. The so-called Variational Garrote stands out by providing a simple yet, principled approach for direct subset feature selection based on a variational approximation to the posterior distribution over subsets [5], see also the related work in [6]. In the original formulation the Variational Garrote is based on two hyperparameters representing a prior belief in sparsity and the noise level, respectively, and it was proposed to cross-validate the former while the second was inferred [5].

Here we make two contributions to improve our understanding of the Variational Garrote and hopefully make it even more useful in practical applications. The first contribution is a simple reformulation of the sparsity promoting prior, the proposed form involves a hyperparameter which can be directly interpreted as the prior sparsity rate, and furthermore leads to very a simple update equation if we attempt to make inference for hyperparameter. The second contribution is an evaluation of the Variational Garrote with inferred and cross-validated hyperparameters. In particular, we compare with the convex relaxation approach LASSO and with the ARD approach for direct solution of the combinational problem. We find empirical evidence that there is a significant range of hyperparameters for which ARD and the Variational Garotte can find near-optimal solutions, both in terms of mean square test error, and in terms of location of the non-zero variables.

2. THE VARIATIONAL GARROTE

We study a general linear regression setting. Let $\mathcal{D} = \{\mathbf{x}^\mu, y^\mu | \mu = 1...p\}$ be a data set, where $\mathbf{x}^\mu$ is an n-dimensional feature vector and y^μ is a scalar response variable to be modeled. The Variational Garrote generative model as introduced in [5] is then given by

$$y^\mu = \sum_{i=1}^{n} w_i s_i \mathbf{x}_i^\mu + \xi^\mu \tag{1}$$

where w_i is the i'th weight and ξ^μ for $\mu = 1..p$ are assumed to be independent and normal distributed with zero mean and variance $\sigma^2 = \beta^{-1}$, i.e. $\xi \sim \mathcal{N}\left(0, \beta^{-1}\right)$. The variables $s_i \in$

This work is funded in part by the Danish Lundbeckfonden through the Center for Integrated Molecular Brain Imaging (www.cimbi.dk).

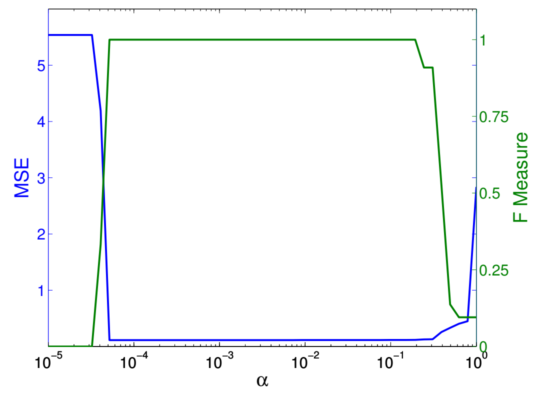


Fig. 1. Variational Garrote: Mean square error and F-measure as a function of α. Inference is based on $p = 50$ samples from the linear model $y = \mathbf{w}^T\mathbf{x} + \xi$, where $\mathbf{x}$ is an $N = 100$ dimensional feature vector sampled from a Normal distribution ($\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$). The true weights $\mathbf{w}$ are given by $\mathbf{w} = \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 0 & \dots & 0\end{bmatrix}^T \in \mathbb{R}^{100}$ and $\xi \sim \mathcal{N}(0, 0.1)$. The test set comprised $p_{test} = 400$ samples.

$\{0, 1\}$ are latent binary selector variables, which determine the set of features used in the model. That is, the i'th weight is removed from the model when $s_i = 0$. The prior distribution over the latent variable $\mathbf{s}$ proposed by [5] is

$$p(\mathbf{s}|\gamma) = \prod_{i=1}^{n} p(s_i|\gamma) \quad \text{with} \quad p(s_i|\gamma) = \frac{\exp(\gamma s_i)}{1 + \exp(\gamma)}. \quad (2)$$

In this representation $\gamma < 0$ implies that the prior belief in $s_i = 1$ is below 0.5 and therefore the model is biased towards a sparse solution.

The joint posterior distribution of $\mathbf{s}$, $\mathbf{w}$ and β conditioned on the data set $\mathcal{D}$ is then,

$$P(\mathbf{s}, \mathbf{w}, \beta|\mathcal{D}, \gamma) = \frac{P(\mathcal{D}|\mathbf{s}, \mathbf{w}, \beta) P(\mathbf{w}, \beta) P(\mathbf{s}|\gamma)}{P(\mathcal{D}|\gamma)} \quad (3)$$

where $P(\mathbf{w}, \beta)$ is the prior distribution over the weights and the precision parameter and is for simplicity assumed to be an improper uniform distribution, i.e., $P(\mathbf{w}, \beta) \propto 1$. Furthermore, we note that the denominator of the right hand side, $P(\mathcal{D}|\gamma)$, does not depend on the parameters,

$$P(\mathbf{s}, \mathbf{w}, \beta|\mathcal{D}, \gamma) \propto P(\mathcal{D}|\mathbf{s}, \mathbf{w}, \beta) P(\mathbf{s}|\gamma) \quad (4)$$

Since the posterior distribution of $\mathbf{w}$ and β conditioned on the data is our main interest, we follow the Bayesian paradigm and marginalize out the $\mathbf{s}$ variable

$$P(\mathbf{w}, \beta|\mathcal{D}, \gamma) \propto \sum_{\mathbf{s}} P(\mathcal{D}|\mathbf{s}, \mathbf{w}, \beta) P(\mathbf{s}|\gamma). \quad (5)$$

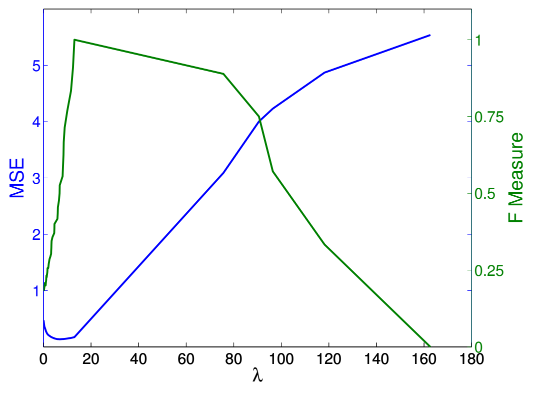


Fig. 2. LASSO: Mean square error and F-measure as a function of λ. Estimation is based on $p = 50$ samples from the linear model $y = \mathbf{w}^T\mathbf{x} + \xi$, where $\mathbf{x}$ is an $N = 100$ dimensional feature vector sampled from a Normal distribution ($\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$). The true weights $\mathbf{w}$ are given by $\mathbf{w} = \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 0 & \dots & 0\end{bmatrix}^T \in \mathbb{R}^{100}$ and $\xi \sim \mathcal{N}(0, 0.1)$. The test set comprised $p_{test} = 400$ samples. We used the toolbox implemented by Sjostrand to estimate the LASSO [7]

Exact inference is infeasible due to the discrete nature of $\mathbf{s}$ (involves averaging over 2^n binary configurations) and therefore we resort to an approximation.

In particular, we apply a variational approximation introducing a fully factorized variational distribution over $\mathbf{s}$. That is,

$$q(\mathbf{s}) = \prod_{i=1}^{n} q_i(s_i) \quad (6)$$

where $q_i(s_i) = m_i s_i + (1 - m_i)(1 - s_i)$. By Jensen's inequality [8], we obtain

$$\begin{aligned} & \ln\left(P(\mathcal{D}|\mathbf{s}, \mathbf{w}, \beta) P(\mathbf{s}|\gamma)\right) \\ & \geq -\sum_{\mathbf{s}} q(\mathbf{s}) \ln\left(\frac{q(\mathbf{s})}{P(\mathcal{D}|\mathbf{s}, \mathbf{w}, \beta) P(\mathbf{s}|\gamma)}\right) \\ & \equiv -F(q, \mathbf{w}, \beta), \end{aligned}$$

where $F(q, \mathbf{w}, \beta)$ is the variational free energy. Minimizing the variational free energy (hence, maximizing the lower bound on the marginal likelihood) yields the following update equations [5]:

$$\mathbf{w} = (\chi')^{-1}\mathbf{b}, \qquad \chi'_{ji} = m_i \chi_{ji} + (1 - m_i)\chi_{jj}\delta_{ij} \quad (7)$$

$$m_i = \sigma\left(\gamma + \frac{\beta p}{2}\chi_{ii} w_i^2\right) \quad (8)$$

$$\frac{1}{\beta} = \sigma_y^2 - \sum_i m_i w_i b_i, \quad (9)$$

where χ is the second moment matrix of the input features, $\sigma()$ is the sigmoid function, and $\mathbf{b}$ is the input-output covariance.

2.1. Re-parametrization of the prior

The prior distribution on $\mathbf{s}$ in (2) is simply a product of independent Bernoulli distributions, where the corresponding probability of $s_i = 1$ is parameterized by γ:

$$p\left(s_i = 1|\gamma\right) = \frac{\exp\left(\gamma\right)}{1 + \exp(\gamma)} \tag{10}$$

Due to the functional relationship, the interpretation of γ is less transparent and inference can lead to complex non-linear equations for the parameter. Therefore we propose a more simple parametrization of the prior distribution given by,

$$p\left(s_i|\alpha\right) = \alpha s_i + (1 - \alpha)\left(1 - s_i\right), \qquad \alpha \in [0, 1], \tag{11}$$

see also the different but related formulation in [6]. Hence, $p\left(s_i = 1|\alpha\right) = \alpha$. Thus, $p\left(\mathbf{s}|\alpha\right)$ becomes a Binomial distribution. As there is a close correspondence between the two priors

$$\gamma = \ln\left(\frac{\alpha}{1 - \alpha}\right), \tag{12}$$

the new parametrization only implies minor changes in the variational approximation, e.g., the update equation for the variational means becomes

$$m_i = \sigma\left(\ln\left(\frac{\alpha}{1 - \alpha}\right) + \frac{\beta p}{2}\chi_{ii} w_i^2\right) \tag{13}$$

2.2. Learning α

Since α is a probability, we naturally assign a Beta-distribution as the prior distribution over α

$$P\left(\alpha|a, b\right) = \frac{\Gamma\left(a + b\right)}{\Gamma\left(a\right)\Gamma\left(b\right)}\alpha^{a-1}\left(1 - \alpha\right)^{b-1} \tag{14}$$

where a and b are hyper-hyperparameters.

With this assignment the posterior becomes

$$P\left(\mathbf{w}, \alpha, \beta|\mathcal{D}\right) \propto \sum_s P\left(\mathcal{D}|\mathbf{s}, \mathbf{w}, \beta\right) P\left(\mathbf{s}|\alpha\right) P\left(\alpha\right) \tag{15}$$

and the variational free energy becomes

$$F\left(q, \mathbf{w}, \alpha, \beta\right) = \sum_{\mathbf{s}} q\left(\mathbf{s}\right) \ln\left(\frac{q\left(\mathbf{s}\right)}{P\left(\mathcal{D}|\mathbf{s}, \mathbf{w}, \beta\right) P\left(\mathbf{s}|\alpha\right) P\left(\alpha\right)}\right) \tag{16}$$

Minimizing the variational free energy w.r.t. α yields an intuitive update rule,

$$\alpha = \frac{\sum_i^n m_i + a - 1}{n + a + b - 2} \tag{17}$$

As $\sum_i^n m_i$ is the expected number of non-zero weights the two hyper-hyperparameters $a-1$ and $b-1$ can be interpreted as pseudo-observations (beta-binomial model), such that $a-1$ and $b-1$ are pseudo-counts of non-zero and zero weights, respectively.

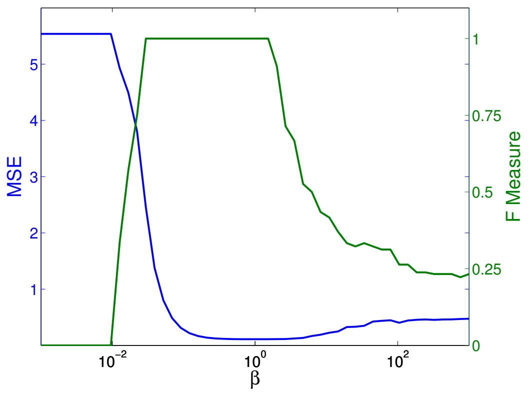


Fig. 3. ARD: Mean square error and F-measure as a function of β. Inference is based on $p = 50$ samples from the linear model $y = \mathbf{w}^T\mathbf{x} + \xi$, where $\mathbf{x}$ is an $N = 100$ dimensional feature vector sampled from a Normal distribution ($\mathbf{x} \sim \mathcal{N}_{100}\left(\mathbf{0}, \mathbf{I}\right)$). The true weights $\mathbf{w}$ are given by $\mathbf{w} = \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 0 & \ldots & 0\end{bmatrix}^T \in \mathbb{R}^{100}$ and $\xi \sim \mathcal{N}\left(0, 0.1\right)$. The test set comprised $p_{test} = 400$ samples. For details of the ARD inference procedure see [8]

3. NUMERICAL EXPERIMENTS

To illustrate the role of the reformulated prior we carry out three experiments: the first comparing the Variational Garrote with two widely used standard methods as in the original work [5] and in the second we investigate the feasibility of learning the hyperparameter α and compare it with cross-validation over α and γ respectively. Finally, in a third experiment we make a more complete exploration of the connection between number of samples, sparsity of the ”true” weight vector used to generate data, and performance.

3.1. Experiment 1

In the first experiment, we compare the Variational Garrote to the LASSO [1] and to a sparse linear regression model using an ARD prior [3, 9, 4]. Both of these methods have a single hyperparameter (the penalty control λ for LASSO and the

[2]Figure 4 is slightly different from the corresponding figure in the published version. This is due to an minor scaling error resulting in worse performance of the two CV-based methods.

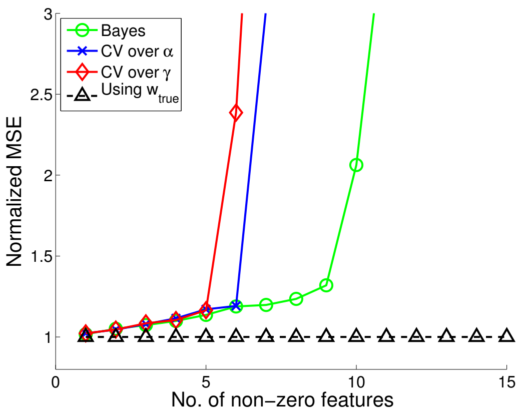


Fig. 4. Normalized mean square error as a function of number of non-zero features. The features $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$ and the noise $\xi \sim \mathcal{N}(0, \sigma_0^2)$ are both normal. Here we vary the number of non-zero coefficients in the true model's weights $\mathbf{w}$. The magnitudes of all non-zero coefficients are fixed to value 1, while the true noise variance σ_0^2 is scaled to provide a fixed signal-to-noise ratio as we increase the number of true non-zero parameters. We compare three models, namely the Variational Garrote with the conventional parametrization γ and the new α both estimated by hold-out cross-validation and the Variational Garrote with α inferred from data with a Beta(1,99) hyper-prior. All results are averaged over 100 runs, $n = 100$ $p = 50$, $p_{\text{test}} = 400$. [2]

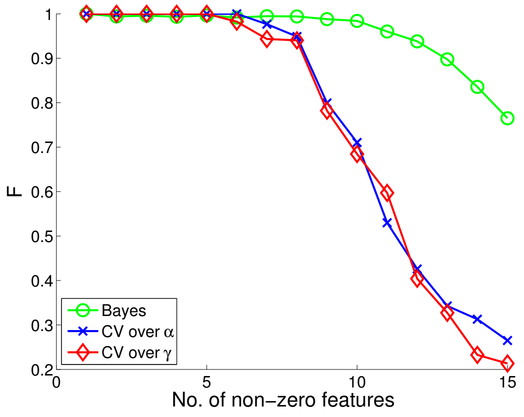


Fig. 5. F-measure as a function of number of non-zero features. The features $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$ and the noise $\xi \sim \mathcal{N}(0, \sigma_0^2)$ are both normal. Here we vary the number of non-zero coefficients in the true model's weights $\mathbf{w}$. The magnitudes of all non-zero coefficients are fixed to value 1, while the true noise variance σ_0^2 is scaled to provide a fixed signal-to-noise ratio as we increase the number of true non-zero parameters. We compare three models, namely the Variational Garrote with the conventional parametrization γ and the new α both estimated by hold-out cross-validation and the Variational Garrote with α inferred from data with a Beta(1,99) hyper-prior. All results are averaged over 100 runs, $n = 100$, and $p = 50$.

noise precision β for ARD) and we can compare the performance as we vary these parameter with the results of varying the prior parameter α for VG, The three methods are compared by means of test set Mean Square Error (MSE) and the F-measure of the feature selection process given as [10],

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{18}$$

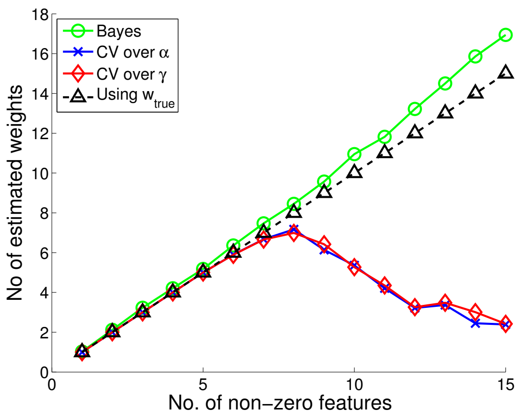
where precision (positive predictive value) is the fraction of non-zero weights found by the algorithm that are also non-zero in the true model, while recall (sensitivity) is the fraction of non-zeros in the true model that have been identified by the algorithm.

We generate $p = 50$ samples from the model $y = \mathbf{w}^T\mathbf{x} + \xi$, where $\mathbf{x}$ is an $N = 100$ dimensional feature vector sampled from an isotropic multivariate Normal distribution, i.e. $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$, the weights $\mathbf{w}$ are given by $\mathbf{w} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & \ldots & 0 \end{bmatrix}^T \in \mathbb{R}^{100}$ and $\xi \sim \mathcal{N}(0, 0.1)$. Furthermore, we generate an additional $p_{test} = 400$ samples from the same model, which is used as an independent test set. For each method and for each value of their hyperparameters, the mean square test error and F-measure are computed.

Figures 1, 2 and 3 show the resulting plots for VG, LASSO and ARD, respectively. Inspecting Figure 2 we see that both the Mean Square Error and the F-measure are critically dependent on the strength of the regularizer λ, indeed there is a narrow interval around $\lambda = 10$ where both the Mean Square Error is low and the F-measure is high, but no value for which they both are optimal. On the other hand we learn from Figure 3 that in the ARD there are about two decades of values of the noise precision for which the F-measure reaches its optimal value (F=1) and for a significant part of this range the Mean Square Error on the test data is simultaneously minimized. Similarly for the Variational Garrote we see in Figure 1 that there there is an even wider range of almost four decades of the hyperparameter for which the correct model is identified, hence F=1 and the test Mean Square Error is low. This result is promising, and lead us to hypothesize that it may be possible to infer the hyperparameter α from data.

3.2. Experiment 2

The second experiment is designed precisely to investigate the possibility of learning hyperparameter α directly from the data. We assign a Beta-distribution as given in (14) with pa-



(c) Number of estimated weights

Fig. 6. Number of estimated components as a function of non-zero features. The features $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$ and the noise $\xi \sim \mathcal{N}\left(0, \sigma_0^2\right)$ are both normal. Here we vary the number of non-zero coefficients in the true model's weights $\mathbf{w}$. The magnitudes of all non-zero coefficients are fixed to value 1, while the true noise variance σ_0^2 is scaled to provide a fixed signal-to-noise ratio as we increase the number of true non-zero parameters. We compare three models, namely the Variational Garrote with the conventional parametrization γ and the new α both estimated by hold-out cross-validation and the Variational Garrote with α inferred from data with a Beta(1,99) hyper-prior. All results are averaged over 100 runs, $n = 100$, and $p = 50$.

rameters $a = 1$ and $b = 99$, i.e., invoking a (hyper-)prior assuming strong sparsity. We generate a fixed number $p = 50$ samples from the model $y = \mathbf{w}^T\mathbf{x} + \xi$. The features $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$ and the noise $\xi \sim \mathcal{N}\left(0, \sigma_0^2\right)$ are both normal as in experiment 1, but now we vary the number of non-zero coefficients in the model weights $\mathbf{w}$. The magnitudes of all non-zero coefficients are again fixed to value 1, while the true noise variance σ_0^2 is scaled to provide a fixed signal-to-noise as we increase the number of true non-zero parameters. We compare three models, namely the Variational Garrote with the conventional parametrization γ and the new α both estimated by hold-out cross-validation and the Variational Garrote with α inferred from data. All results are average over 100 runs.

The test Mean Square Error normalized by the noise variance is presented as a function of non-zero weights in figure 4 for the three models, for further reference we also plot the error of the true solution. The Bayesian model outperforms both cross-validated models as hypothesized since the hyper-prior conforms with the sparse data generating true model [4, 11]. In addition we see that the new prior formulation seems to

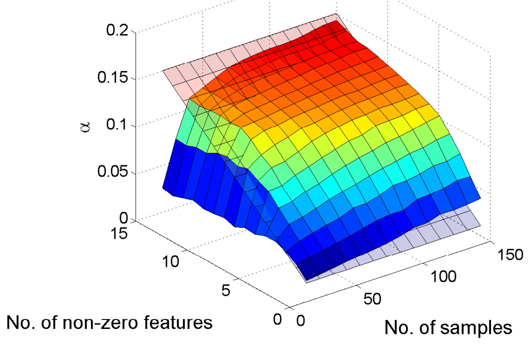


Fig. 7. α inferred from the data as a function of number of non-zero features and no. of samples. Samples are generated from the model $y = \mathbf{w}^T\mathbf{x} + \xi$. The features $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$ and the noise $\xi \sim \mathcal{N}\left(0, \sigma_0^2\right)$. But now we vary both the number of non-zero coefficients in the model and the no. of samples n used to estimate the model. The magnitudes of all non-zero coefficients are still fixed to value 1, while the true noise variance σ_0^2 is scaled to provide a fixed signal-to-noise as we increase the number of true non-zero parameters.

provide the best performance of the two cross-validated models when more than five of the hundred parameters in the true model are non-zero.

In Figure 5 we show the F-measure for the three models as the true model becomes denser. Here we find that all three models have close to optimal retrieval of the true locations up till around seven of the hundred weights are non-zero. The cross-validated γ model seems to be bit more stable. However, as the true models become denser only the Bayesian inference model can maintain high F, while in the range 7-14 non-zero weights the cross-validated α model outperforms the γ model significantly.

The results of the Figures 4 and 5 can in part be understood when we plot in Figure 6 the number of estimated weights as a function of number of non-zero features. Here we learn that the cross-validated models focusing on MSE become too conservative and thereby under-estimate the number of non-zero weights.

3.3. Experiment 3

The aim of the third experiment is to investigate further the properties of the Bayesian model. We generate n samples from the model $y = \mathbf{w}^T\mathbf{x} + \xi$. The features $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$ and the noise $\xi \sim \mathcal{N}\left(0, \sigma_0^2\right)$. But now we vary both the number of non-zero coefficients in the model and the number of samples n to explore learnability more generally. The mag-

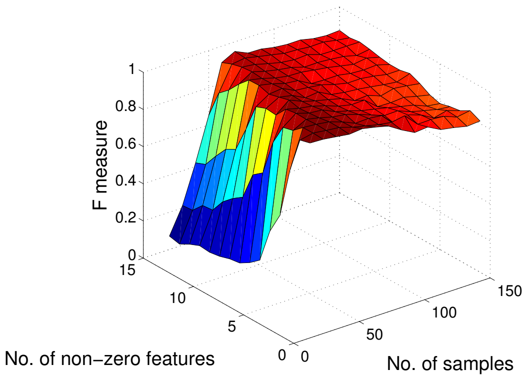


Fig. 8. F-measure as a function of number of non-zero features and no. of samples. Samples are generated from the model $y = \mathbf{w}^T\mathbf{x} + \xi$. The features $\mathbf{x} \sim \mathcal{N}_{100}(\mathbf{0}, \mathbf{I})$ and the noise $\xi \sim \mathcal{N}\left(0, \sigma_0^2\right)$. But now we vary both the number of non-zero coefficients in the model and the no. of samples n used to estimate the model. The magnitudes of all non-zero coefficients are still fixed to value 1, while the true noise variance σ_0^2 is scaled to provide a fixed signal-to-noise as we increase the number of true non-zero parameters.

nitudes of all non-zero coefficients are still fixed to value 1, while the true noise variance σ_0^2 is scaled to provide a fixed signal-to-noise as we increase the number of true non-zero parameters.

Figure 7 shows the value of α inferred from the data as a function of the number of non-zero coefficients and sample size, while in Figure 8 we show the corresponding F-measures, which reveals that very accurate feature locations are found for a sample size abaout three times the number of true non-zero parameters, while still much smaller than the number of features $p < n$.

4. CONCLUSION

We reformulated the prior of the Variational Garrote to follow a simple Binomial law and assigned a Beta hyper-prior on the parameter. With the new prior the Variational Garrote has a wide range of parameter values for which it at the same time provides for low test error and high retrieval of the true feature locations. Furthermore, the new prior formulation leads to a simple update rule in Bayesian variational inference for the hyperparameter. In a second simulation study we provided evidence that the new procedure can improve on cross-validation of the parameters. We also found that the new formulation of the prior outperforms the original formulation when both are cross-validated to determine their respective hyperparameters. Finally, we found that the Bayesian scheme produces very accurate feature locations when the number of true non-zeros is small, allowing for good performance even when the system formally is ill-posed $p < n$.

5. REFERENCES

[1] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[2] N. Meinshausen and P. Bühlmann, “Stability selection,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 4, pp. 417–473, 2010.

[3] R. M. Neal, *Bayesian learning for neural networks*, Ph.D. thesis, University of Toronto, 1995.

[4] L. K. Hansen and C. E. Rasmussen, “Pruning from adaptive regularization,” *Neural Computation*, vol. 6, no. 6, pp. 1223–1232, 1994.

[5] H. J. Kappen and V. Gómez, “The Variational Garrote,” *ArXiv e-prints*, Sept. 2011.

[6] M. Lzaro-gredilla and M. K. Titsias, “Spike and slab variational inference for multi-task and multiple kernel learning,” in *Advances in Neural Information Processing Systems 24*, J. Shawe-taylor, R.s. Zemel, P. Bartlett, F.c.n. Pereira, and K.q. Weinberger, Eds., pp. 2339–2347. 2011.

[7] K. Sjöstrand, “Matlab implementation of LASSO, LARS, the elastic net and SPCA,” jun 2005, Version 2.0.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] D. MacKay, “Comparison of approximate methods for handling hyperparameters,” *Neural computation*, vol. 11, no. 5, pp. 1035–1068, 1999.

[10] C. J. V. Rijsbergen, *Information Retrieval*, Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[11] L. K. Hansen, “Bayesian averaging is well-temperated,” *Advances in Neural Information Processing Systems*, 1999.

Bibliography

[AHH13] Michael Riis Andersen, Sofie Therese Hansen, and Lars Kai Hansen. Learning the solution sparsity of an ill-posed linear inverse problem with the variational garrote. *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2013.

[AM12] Javier M. Antelis and Javier Minguez. Eeg source localization based on dynamic bayesian estimation techniques. 2012.

[AS72] DL Alspach and HW Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, AC17(4):439–&, 1972.

[Bar11] D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 04-2011 edition, 2011. In press.

[BG97] S. Baillet and L. Garnero. A bayesian approach to introducing anatomo-functional priors in the EEG/MEG inverse problem. *Biomedical Engineering*, 44(5):374–385, May 1997.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BLP$^+$12] B. Babadi, C. Lamus, P. L. Purdon, E. N. Brown, and E. Pirondini. A spatially-regularized dynamic source localization algorithm for eeg. *Conference Proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*

IEEE Engineering in Medicine and Biology Society. Conference, 2012:6752–6755, 2012.

[BM10] Mohsen Bayati and Andrea Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. In *ISIT*, pages 1528–1532. IEEE, 2010.

[BML01] S. Baillet, J. C. Mosher, and R. M. Leahy. Electromagnetic brain mapping. *Signal Processing Magazine, IEEE*, 18(6):14–30, November 2001.

[BTA11] Jie Yang 0009, Abdesselam Bouzerdoum, Fok Hing Chi Tivive, and Moeness G. Amin. Multiple-measurement vector model and its application to through-the-wall radar imaging. In *ICASSP*, pages 2672–2675. IEEE, 2011.

[CDA08] Scott Shaobing Chen, David L. Donoho, and Michael A. Atomic decomposition by basis pursuit. 2008.

[CP91] SD Cabrera and TW Parks. Extrapolation and spectral estimation with iterative weighted norm modification. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 39(4):842–851, 1991.

[CREKD05] SF Cotter, BD Rao, K. Engan, and K. Kreutz-Delgado. Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 53(7):2477–2488, 2005.

[CRT06] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[DLR11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. 2011.

[DMA97] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *CONSTRUCTIVE APPROXIMATION*, 13(1):57–98, 1997.

[DMM09] David L. Donoho, A. Maleki, and A. Montanari. Message-passing algorithms for compressed sensing. *Proc. Nat. Acad. Sci. U.S.A.*, 106(45):18914–18919, 2009.

[DMM10] David L Donoho, Arian Maleki, and Andrea Montanari. How to design message passing algorithms for compressed sensing, 2010.

[DMM11] David L. Donoho, Arian Maleki, and Andrea Montanari. The noise-sensitivity phase transition in compressed sensing. *IEEE Transactions on Information Theory*, 57(10):6920–6941, 2011.

[DT09] D. Donoho and J. Tanner. Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. *PHILOSOPHICAL TRANSACTIONS- ROYAL SOCIETY OF LONDON SERIES a MATHEMATICAL PHYSICAL AND ENGINEERING SCIENCES*, 367(1906):4273–4294, 2009.

[DT10] David L. Donoho and Jared Tanner. Precise undersampling theorems. *Proceedings of the IEEE*, 98(6):913–924, 2010.

[Dur04] Richard Durrett. *Probability: Theory and Examples (Probability: Theory & Examples)*. Duxbury Press, 3 edition, March 2004.

[EHJT] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Ann. Statist*, page 2004.

[EK12] Yonina C. Eldar and Gitta Kutyniok. *Compressed sensing : theory and applications*. Cambridge University Press, 2012.

[Ela12] Michael Elad. Sparse and redundant representation modeling-what next? *IEEE SIGNAL PROCESSING LETTERS*, 19(12):922–928, 2012.

[FM98] Brendan Frey and David MacKay. A revolution: Belief propagation in graphs with cycles. In *In Neural Information Processing Systems*, pages 479–485. MIT Press, 1998.

[GKH12] Alexandre Gramfort, Matthieu Kowalski, and Matti Haemaelaeinen. Mixed-norm estimates for the m/eeg inverse problem using accelerated gradient methods. *PHYSICS IN MEDICINE AND BIOLOGY*, 57(7):1937–1961, 2012.

[GPOC10] Alexandre Gramfort, Theodore Papadopoulo, Emmanuel Olivi, and Maureen Clerc. Openmeeg: opensource software for quasistatic bioelectromagnetics. *BIOMEDICAL ENGINEERING ONLINE*, 9:–, 2010.

[GR97] Irina F. Gorodnitsky and Bhaskar D. Rao. Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, 1997.

[GYO$^+$04] A. Galka, O. Yamashita, T. Ozaki, R. Biscay, and P. Valdes-Sosa. A solution to the dynamical inverse problem of eeg generation using spatiotemporal kalman filtering. *NEUROIMAGE*, 23(2):435–453, 2004.

[HSH13] Sofie Therese Hansen, Carsten Stahlhut, and Lars Kai Hansen. Sparse source eeg imaging with the variational garrote. 2013.

[HVG$^+$07] Hans Hallez, Bart Vanrumste, Roberta Grech, Joseph Muscat, Wim De Clercq, Anneleen Vergult, Yves D'Asseler, Kenneth P. Camilleri, Simon G. Fabri, Sabine Van Huffel, and Ignace Lemahieu. Review on solving the forward problem in eeg source analysis. *Journal of Neuroengineering and Rehabilitation*, 4:46, 2007.

[IR05] Hemant Ishwaran and J. Sunil Rao. Spike and slab variable selection: Frequentist and bayesian strategies. 2005.

[KFL01] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 47(PART 2):498–519, 2001.

[KG12] Hilbert J. Kappen and Vicenç Gómez. The variational garrote. page 26, 2012.

[LDH$^+$07] Hans-Andrea Loeliger, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R. Kschischang. The factor graph approach to model-based signal processing. *PROCEEDINGS OF THE IEEE*, 95(6):1295–1322, 2007.

[Loe04] H.-A. Loeliger. An introduction to factor graphs. *IEEE SIGNAL PROCESSING MAGAZINE*, 21(PART 1):28–41, 2004.

[Mac03] David J. C. Mackay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, first edition edition, June 2003.

[Mal13] Arian Maleki. Private communication, December 2013.

[McD09] Gary C. McDonald. Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):93–100, 2009.

[MD10] Arian Maleki and David L. Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *J. Sel. Topics Signal Processing*, 4(2):330–341, 2010.

[Mic09] Christoph M. Michel. *Electrical neuroimaging.* Cambridge University Press, 2009.

[MLL99] J. C. Mosher, R. M. Leahy, and P. S. Lewis. Eeg and meg: Forward solutions for inverse methods. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING BME*, 46(3):245–259, 1999.

[MMC09] Robert J. Mceliece, David J. C. Mackay, and Jung-fu Cheng. Turbo decoding as an instance of pearl's belief propagation algorithm. 2009.

[MT11] Henrik. Madsen and Poul. Thyregod. *Introduction to general and generalized linear models*. CRC Press, 2011.

[MW12] Angshul Majumdar and Rabab K. Ward. Face recognition from video: An mmv recovery approach. In *ICASSP*, pages 2221–2224. IEEE, 2012.

[MWJ99] KP Murphy, Y. Weiss, and MI Jordan. Loopy belief propagation for approximate inference: An empirical study. *UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, PROCEEDINGS*, pages 467–475, 1999.

[Nea95] Radford M. Neal. *BAYESIAN LEARNING FOR NEURAL NETWORKS*. PhD thesis, University of Toronto, 1995.

[NH98] RM Neal and GE Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. *LEARNING IN GRAPHICAL MODELS*, 89:355–368, 1998.

[NS06] Paul L. Nunez and Ramesh. Srinivasan. *Electric fields of the brain : The neurophysics of EEG*. Oxford University Press, 2006.

[NT10] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. 2010.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

[OBDF09] J. P. Oliveira, J. M. Bioucas-Dias, and M. A. Figueiredo. Adaptive total variation image deblurring: A majorization-minimization approach. *SIGNAL PROCESSING -AMSTERDAM-*, 89(9):1683–1693, 2009.

[OFMS11] Robert Oostenveld, Pascal Fries, Eric Maris, and Jan-Mathijs Schoffelen. Fieldtrip: Open source software for advanced analysis of meg, eeg, and invasive electrophysiological data. *Intell. Neuroscience*, 2011:1:1–1:9, January 2011.

[OPD$^+$12] Julie Onton, Jason Palmer, Arnaud Delorme, Scott Makeig, and Robert Oostenveld. Independent eeg sources are dipolar. *PLoS ONE*, 7(2):–, 2012.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[PHC08] Chris Plummer, A. Simon Harvey, and Mark Cook. Eeg source localization in focal epilepsy: Where are we now ? *EPILEPSIA*, 49(2):201–218, 2008.

[PK11] Mark A. Pinsky and Samuel. Karlin. *An introduction to stochastic modeling.* Academic Press, 2011.

[Ran10] Sundeep Rangan. Generalized approximate message passing for estimation with random linear mixing. *CoRR*, abs/1010.5141, 2010.

[REC$^+$10] Bhaskar D. Rao, Kjersti Engan, Shane F. Cotter, Jason Palmer, and Kenneth Kreutz-delgado. Subset selection in noise based on diversity measure minimization. 2010.

[Rij79] C. J. Van Rijsbergen. *Information Retrieval.* Butterworth, 1979.

[RJ86] Lawrence R. Rabiner and Biing-Hwang Juang. Introduction to hidden markov models. *IEEE ASSP Magazine (Acoustics, Speech, and Signal Processing)*, 3(1):4–16, 1986.

[RS08] Alejandro Ribes and Francis Schmitt. Linear inverse problems in imaging. *IEEE SIGNAL PROCESSING MAGAZINE*, 25(4):84–99, 2008.

[RZ96] J. B. Ramsey and Z. Zhang. The application of wave form dictionaries to stock market index data. *SPRINGER SERIES IN SYNERGETICS*, 69:189–208, 1996.

[SAS$^+$13] Carsten Stahlhut, Hagai T. Attias, Kensuke Sekihara, David Wipf, Lars Kai Hansen, and Srikantan S. Nagarajan. How about a bayesian m/eeg imaging method correcting for incomplete spatio-temporal priors. *Journal of Cognitive Neuroscience*, Supplement:260, 2013.

[Sch10] Philip Schniter. Turbo reconstruction of structured sparse signals. In *in Proc. 44th Annual Conf. Information Sciences and Systems*, 2010.

[SFM10] Jean-Luc Starck, Jalal M. Fadili, and Fionn Murtagh. *Sparse image and signal processing : wavelets, curvelets, morphological diversity.* Cambridge University Press, 2010.

[Sjö05] K. Sjöstrand. Matlab implementation of LASSO, LARS, the elastic net and SPCA, jun 2005. Version 2.0.

[Sta08] Carsten Stahlhut. EEG source localization using a hierarchical bayesian approach. Master's thesis, Technical University of Denmark, 2008.

[TA77] A.N. Tikhonov and V.Y. Arsenin. *Solutions of ill-posed problems.* Wiley, 1977.

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[Tip01] ME Tipping. Sparse bayesian learning and the relevance vector machine. *JOURNAL OF MACHINE LEARNING RESEARCH*, 1(3):211–244, 2001.

[TSK06] Pang-Ning. Tan, Michael. Steinbach, and Vipin. Kumar. *Introduction to data mining.* Pearson, Addison Wesley, 2006.

[vdBF10] Ewout van den Berg and Michael P. Friedlander. Theoretical and empirical results for recovery from multiple measurements. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 56(5):2516–2527, 2010.

[VS13] Jeremy P. Vila and Philip Schniter. Expectation-maximization gaussian-mixture approximate message passing. *IEEE Transactions on Signal Processing*, 61(19):4658–4672, 2013.

[WN09] David Wipf and Srikantan Nagarajan. A new view of automatic relevance determination. 2009.

[WR07] David P. Wipf and Bhaskar D. Rao. An empirical bayesian strategy for solving the, simultaneous sparse approximation problem. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 55(7):3704–3716, 2007.

[ZR13] Zhilin Zhang and Bhaskar D. Rao. Sparse signal recovery with temporally correlated source vectors using sparse bayesian learning. 2013.

[ZRM09] R. K. P. Zia, Edward F. Redish, and Susan R. Mckay. Making sense of the legendre transform. *AMERICAN JOURNAL OF PHYSICS*, 77(7):614–622, 2009.

[ZRZ$^+$08] Lun Zhao, Qiushi Ren, Yisheng Zhu, Yingjie Li, and Xiaoyan Du. Removal of artifacts from eeg signal. *Sheng Wu Yi Xue Gong Cheng Xue Za Zhi = Journal of Biomedical Engineering = Shengwu Yixue Gongchengxue Zazhi*, 25(2):464–467, 471, 2008.

[ZS13a] Justin Ziniel and Philip Schniter. Dynamic compressive sensing of time-varying signals via approximate message passing. *IEEE Transactions on Signal Processing*, 61(21):5270–5284, 2013.

[ZS13b] Justin Ziniel and Philip Schniter. Efficient high-dimensional inference in the multiple measurement vector problem. *IEEE Transactions on Signal Processing*, 61(2):340–354, 2013.

[ZS14] J. Ziniel and P. Schniter. Binary linear classification and feature selection via generalized approximate message passing. *ArXiv e-prints*, January 2014.

[ZWF01] H. Zhou, X. Wu, and H. Feng. Independent component analysis and its application for preprocessing eeg. *Beijing Shengwu Yixue Gongcheng/Beijing Biomedical Engineering*, 20(1):35–46, 2001.